GISMO: Competition Results And Final Report

1-1-1992

David R. Van Brackle

Find similar works at: https://stars.library.ucf.edu/istlibrary

University of Central Florida Libraries http://library.ucf.edu

Recommended Citation


This Research Report is brought to you for free and open access by the Digital Collections at STARS. It has been accepted for inclusion in Institute for Simulation and Training by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
GISMO:

COMPETITION RESULTS

AND FINAL REPORT
GISMO
Competition Results and Final Report

Contract N61339-89-C-0044

February 28, 1992

IST-TR-92-15

Prepared by
David Van Brackle

Reviewed by
Curtis Lisle
GISMO: Competition Results and Final Report

David R. Van Brackle

February 28, 1992

IST-TR-92-9

Institute for Simulation and Training
12424 Research Parkway, Suite 300
Orlando, FL 32816-0544
1 Purpose

This technical report is submitted as a deliverable item Number 2B, specified under Task 4, "Benchmark Development", of Workplan 4, dated 4 January, 1991 for DARPA contract N61339-89-C-0044, INTELLIGENT SIMULATED FORCES: EVALUATION AND EXPLORATION OF COMPUTATIONAL AND HARDWARE STRATEGIES. It discusses GISMO, the Game for Intelligent Simulated Military Opponents, which acts as a testing platform for intelligent simulated force algorithms.

2 Introduction

GISMO is the Game for Intelligent Simulated Military Opponents. It was created as a means of stimulating research into the fields of behavioral modelling of military forces and interactive distributed simulation. Competitors were invited to write programs to control a small force of tanks in a simple scenario. The programs would communicate with a simulator via 2400-baud modems on toll-free telephone numbers, and compete interactively against one another.

GISMO was expected to have three distinct benefits: First, it would generate interest and stimulate research into intelligent simulated force algorithms. Second, it would generate and collect intelligent simulated force algorithms from a variety of sources. Third, it would serve as a testbed and simple benchmarking facility for intelligent simulated force algorithms.

(This document shares some text excerpts with IST Technical Report IST-TR-91-2, titled "GISMO: A Game for Intelligent Simulated Military Opponents". Most of the shared text is contained in section 3 of this document. The text is repeated so as to allow each report to be read without requiring the reader to have access to the other.)

3 Project Overview

The project consisted of designing the competition, soliciting competitors, developing the hardware/software system, holding the competition and summarizing the results.

The game consists of a simplified tank scenario. The tanks are abstractions of real tanks, and are capable of moving, turning their turrets, and firing a weapon. The battlefield is a 256x256 grid with rectangular terrain elements which may be forests, mountains or bodies of water. Each terrain element has an effect on the operability of tanks. The object of the scenario is to locate and destroy the opponent's
Of course, the competing programs must prevent their own blockhouses from being found and destroyed. The terrain is not fixed, but may change from match to match.

The competitors in this competition are computer programs. The competing programs communicate with the simulator via 2400-baud modems on toll-free telephone numbers. Once connection and alignment are established, they each receive (over the modem) a description of the terrain for this particular scenario, and the location of their blockhouse. The battlefield is flat except for rectangular terrain elements of the type mentioned above. After sixty seconds, the competing programs must communicate to the simulator the initial placements of its tanks. From this point on, the competition proceeds in two-second time impulses until it ends. In each two-second time impulse, the simulator sends to each competing program a status message, and the competing programs send to the simulator orders for their tanks. The competition ends when one competing program destroys the opponent's blockhouse.

Any group or individual using any type of computing machinery was allowed to compete. No human intervention is allowed once connection has been established between a competing system and the simulator.

The simulator was developed on an IBM-compatible PC 386 machine. Competitors communicated with the simulator via 2400-baud modems on toll-free telephone numbers.

The competition was held in February, 1992. This gave competitors over a full year to develop their programs. The results are being released in this document.

3.1 Hardware / Software Specifics

The main simulator was implemented on an IBM-compatible PC machine in the C programming language. Two 2400-baud external modems were attached to this machine. The modems were also attached to telephone lines with toll-free numbers.

There was also a smaller version of the simulator. This version was distributed to competitors. A simple competitor was integrated into the simulator in order to fully test the system and to give competitors a 'sparring partner.' This is referred to as the Internal Competitor. Lastly, a Terrain Editor and a Game Log Viewing system were developed. All software was developed in Turbo C 2.0.
3.2 Workplan

The following is a description of the tasks which were necessary to complete GISMO.

TASK 1: Design Competition
The rules of the competition should be simple and fair. Any group or individual using any kind of computing machinery should be allowed to participate. The rules should specify how conflicts are resolved and how an overall winner is chosen. (The rules of the contest are in Appendix 8.3)

The scenario itself must be designed to be simple enough so that all pertinent information may be communicated quickly and easily, while still being complex enough to realistically simulate a tank battle. (A description of the game is in Appendix 8.2)

TASK 2: Develop System
Once the scenario is designed, a simulator for it must be implemented. The hardware and system software must be procured and the simulator software written. (See section 3.1)

TASK 3: Solicit Competitors
Advertisements must be placed in major trade magazines for simulation and artificial intelligence. An information packet must be developed which can be sent to competitors upon expression of their interest. The competitors should be allowed to schedule practice bouts on our simulator in order to debug the communications aspects of their programs.

TASK 4: Hold Competition
The focal point of the project is the competition itself. A pairing scheme must be devised based on the number of competitors. The winner must be determined and announced, and awarded any prizes associated with the competition. (See section 6)

TASK 5: Summarize results
A paper should be written summarizing the results of the competition. It should include information about which teams did well, but more importantly why these programs did well. The body of this paper should be a cogent analysis of the algorithms submitted, how they fared and why they succeeded or failed. (This document is that paper)
3.3 Results

All tasks under the GISMO project are now completed. Notices were placed in many ACM and IEEE journals, and in many simulation and artificial intelligence oriented electronic newsgroups. Over 300 parties responded to notices, and were sent information packets. Of these, 65 returned the enclosed Competitor Information Form, and were sent copies of the GISMO simulator.

The software system developed for the project includes the GISMO simulator, a terrain editor (TED), a game log viewer (VIEWGAME), a template competitor which simply sent and received packets with no logic (SHERMAN), and a limited test version of a simple external competitor (PLUTO).

Unfortunately, only one team produced a complete program: Frank Coyle, an instructor at Southern Methodist University in Dallas, Texas, and two undergraduates. The program is called "Rommel" and is described in this document. The failure of other competitors to complete their programs is attributable to the size of the project and particularly to the lack of prizes. While only one competitor produced a complete program, many worked a great deal on GISMO, and many continue to work on GISMO, intending to use it in their research as a testbed for multiple autonomous agent behavioral modelling algorithms.

The GISMO software included an Internal Competitor (IC) developed at IST by David Van Brackle, Donald Cross and Peter Popovich. Rommel was pitted against the IC a number of times in order to observe its behavior. A description of the internal competitor and of each of the competitions is included in this document.

4 The Internal Competitor

The Internal Competitor (IC) was programmed into the GISMO simulator. Its purpose was to give GISMO participants a worthy competitor to practice against.

4.1 Overall Strategy

The IC has two modes of behavior: Before spotting the opponent's blockhouse, and after. Before spotting the opponent's blockhouse, the IC sends four tanks out to search, and keeps three tanks close to its own blockhouse for defense. The initial placement of the four searching tanks is at the
outer range of placement, and the three defense tanks are adjacent to the blockhouse. After spotting the opponent's blockhouse, all tanks (including the three allocated for defense) are given orders to go to the opponent's blockhouse and destroy it.

4.2 Paths and Important Points

Each tank moves along a path consisting of a list of important points. Each point in the path is attainable from the previous point via direct maneuvering; that is, there can be no intervening obstacles. The points in the path from a source point to a destination point are chosen to minimize the distance to the destination point, and to minimize the distance between point (so that the tanks make short "hops"). Paths are plotted by a Best-First Search algorithm.

Important Points are points which are pivotal in navigation. They occur at the corners of terrain pieces (which are always rectangular in GISMO).

(# marks important points)

Forest: Mountain: Water:

In addition, important points are added at the four corners of any destroyed tank, to avoid collisions with it. Important points which are redundant, inside impassable terrain or outside the battlefield are removed.

In Search Mode (before the opponent's blockhouse has been spotted) the four searching tanks attempt to investigate all of the important points on the battlefield, in the hope that in doing so one of them will see the opponent's blockhouse. Each tank plots a path to the closest important point which has not yet been investigated, or has not yet been assigned to a tank for investigation. As soon as it reaches this point, it plots a path to the next closest uninvestigated and unassigned important point, and so on. When a tank is killed, the important point it is assigned to investigate is marked as unassigned and uninvestigated so that it will be investigated by another tank. When the enemy blockhouse is sighted, all tanks plot and execute a path to it.
4.3 Movement

Tanks move along a path from one important point to the next. The tanks always turn so that they are facing the next important point to which they are to travel. They move at half-speed when close to their source or destination points, and full speed in between. They stop when they arrive at their destination point.

Each important point has a semaphore indicating that some tank is currently heading towards that point. A tank will not start moving towards a point until this semaphore is clear. If a tank is destroyed, the semaphore of the point to which it was heading is cleared.

When moving, a tank checks to see if the next grid location in its path is unoccupied. If it is not, and a collision is imminent, the tank will try to turn away. If this fails, the tank will try to slow down or stop.

4.4 Weapons

When an IC tank cannot see an enemy, it rotates its turret like a radar to try to spot an enemy. It keeps its turret pointing at the corners of the tank rather than straight ahead, back or to the sides. These angles, due to GISMO's definition (See Appendix 8.2), subtend greater areas.

Once an IC tank spots an enemy and that enemy is within range of its weapon, it will fire upon this enemy as often as it can. If there are multiple enemies in its view, it prioritizes them as follows: The blockhouse is always the highest priority. If only enemy tanks are visible, then the closest tank has the highest priority. The IC tank will fire its weapon as often as it can, without regard to its position, speed or remaining ammunition, or to the position or speed of the target.

5 Rommel

The following description of Rommel was provided by Frank Coyle, the system's author:
5.1 Overview of SMU's GISMO Competitor

The GISMO competition provided us with an excellent testbed for exploring several aspects of intelligent systems design. Our long-term objective remains to use GISMO as a testbed for investigating strategies for implementing real-time expert systems. Our plan is to utilize the GISMO simulator to test the effectiveness of our real-time expert system implementation and to extend our explorations into the area of

(1) background planning
(2) meta-level control
(3) distributed problem solving

While our competitor did not make use of an expert system implementation, we feel that our C-based competitor can serve as a benchmark against which to measure the performance of a behaviorally similar rule-based competitor.

Our competitor, ROMMEL, was initially derived from the competitor provided with the GISMO simulator. Our initial efforts involved modification of the Internal Competitor to test some of our ideas. After developing a reasonably performing competitor we began the transformation from Internal to External competitor. A significant amount of time was spent trying to make our own communications code work. The code, written by two students, working on a senior project, was abandoned and replaced by code derived from SHERMAN, written by Don Cross. The conversion from an Internal to External Competitor took longer than anticipated and we only got things really working a few hours before the February 13 deadline. Thus, we were not able to implement several changes that would have improved problems that we saw with ROMMEL. The following sections describe aspects of ROMMEL that distinguishes it from the original Internal Competitor.

5.2 Important Points

We felt that the generation of a set of important points was conceptually an important high level task and that the actual navigation to a given important point was a task to be performed by individual tanks. We took this approach with an eye toward a parallel implementation where a central headquarters would assign destinations and individual tanks would be responsible for navigating to those points.

Our important point generation strategy went through several incarnations and we finally settled upon an approach based on
identifying big rectangles that surrounded the major land masses of the terrain and defining points at the midpoint of each side of the rectangle as important points. Specifically, we used rectangles as defined in a BATTLE.FLD file to:

a) identify all adjacent mountain and water terrain (forest was ignored)

b) determine the smallest enclosing rectangle that encapsulated adjacent mountain and water rectangles

c) determine if any of the rectangles obtained in step b overlapped. If so, we merged the rectangles until none of the rectangles overlapped. This left us with a collection of big rectangles. d) the corners of these rectangles become important points - destinations to send the tanks.

Since each of the resultant rectangles were not overlapping, (we referred to them as "big rectangles"), a path generation algorithm was used to generate a series of points that would lead from any given point on the terrain to a destination point by turning corners around the big rectangles. The algorithm generated the points by moving North, South, East or West in an effort to reduce the difference in the x-direction and y-direction between the start and destination points. When a path was blocked in the x or y direction by an obstruction rectangle, a 90 degree turn was made until the obstruction was resolved (i.e. we turned the corner). Each turning point was pushed on a stack. The collection of turning points became the path of a tank.

Thus, given a tank position and an ultimate destination, we computed a path FROM the destination TO the tank position insuring that for the point at the top of the stack there would be a direct line of sight between it and the tank. Once the path was computed, the point on top of the stack became the tank's DESTINATION (DEXT & DESTY). We essentially reused the code provided with the simulator to navigate the tanks to their destinations and when reaching their destinations, the Path-stack was popped to provide a new destination. When the stack was empty, i.e. the tank had reached it's ultimate destination, the tank rotated its turret to observe the field. After complete rotation, the tank then acquired a new important point, computed a path to it and resumed motion.

We found that while this technique did not produce the most economical path in getting tanks to their destination, it was fairly effective in covering territory composed of
rectangular land masses. We found that the approach was less effective in covering irregular land masses particularly when contiguous land masses needed to be merged to form large non-overlapping rectangles needed for the path generation algorithm. For such terrain, if the blockhouse were located out of sight from the edges of any large rectangle, the tanks would never discover it, since the path computation algorithm would never journey inside a big rectangle. However, it was often the case that the blockhouse would be visible inside a big rectangle allowing the tanks to fire upon it and end the battle.

Another problem with this approach was that tanks were often initially positioned inside a big rectangle, totally confusing the path generation algorithm. To deal with this situation, we modified the path generation algorithm to test if either the tank position or destination was inside a big rectangle and if so, points directly navigable to points inside the rectangle were pushed on the Path Stack while those points outside the rectangle were used for the path computation. The result was a series of points, each with a direct line of sight to the next point so that the original navigation algorithm was able to direct the tank to each point and if necessary adjust the direction if the tank got off course. Thus the big rectangles were used only to generate a series of important points but not used in the actual navigation through the terrain.

5.3 Important Point Assignment.

We initially used the technique of assigning an important point to a tank and reassigning the point if the tank became disabled prior to reaching the point. However, we ran into problems during testing when our competitor ran out of important points. This occurred when the battlefield was composed of very large rectangles that subsumed complex terrain. Under such circumstances, our tanks often had no place else to go and so we added a feature that each tank remembered its initial position and when there were no more important points, each tank would "go home". However, this made for static games and so we changed the important points assignment and simply kept rotating through the set of important points, intending to change this strategy. However, because of time constraints we were not able to change this feature and so our competitor simply cycles through the list of important points.
5.4 Defensive Possibility

We were planning to incorporate a defensive feature into the competitor using the "remembered" initial position of each tank. When the blockhouse "sees" an enemy tank, tanks nearby would return home to defend the blockhouse by plotting a path to their initial positions.

5.5 The Forest Strategy

Because tanks sitting on the edge of a forest are not visible unless moving or firing, we decided to take advantage of this fact in two ways:

1. Tanks traveling to a destination point that find themselves within Forest terrain, monitor the terrain directly in front of them. If the upcoming terrain is either Plain or Water, the tanks set a count-down timer and rotate their turret scanning for enemy tanks until the counter reaches zero. If an enemy tank is seen, the tank remains hidden at the forest edge until either the enemy moves out of view or can be fired upon.

2. To insure that tanks take advantage of this strategic maneuver, we performed a separate forest terrain analysis, grouping contiguous forest rectangles into big forest rectangles. The midpoints of each of the sides of the forest rectangles then became candidates for important points. If these points represented the edge of a forest then the point was added to the important points array. If the point was not the edge of a forest (as could be the case when dealing with a forest rectangle composed of smaller component forest rectangles), points were tested from the edge of the rectangle back toward the center searching for a forest edge. If found, the point was added to the important points array.

This technique proved quite effective in a number of test battles we conducted where a tank at the edge of a forest was able to surprise and kill an enemy tank before the enemy saw it. The technique did not work effectively when an enemy was disabled and out of firing range. Under such a circumstance, our tanks waited forever at the edge of the forest. Fixing this flaw would be a high priority in a revised competitor.
5.6 The Reserve Tanks.

We decided to keep two tanks back at the blockhouse to serve as defenders and as backup should all our other tanks become immobile. The home base tanks were dispatched only when either a) all other tanks were immobile or b) the blockhouse was seen, in which case a path to the blockhouse was computed for the tanks and they were dispatched.

5.7 The Opposite Important Point

A heuristic to second guess the simulated battlefield was added by one of the students working on the project. His suggestion was to have one tank "go long" to a point diagonally opposite the home base under the assumption that the enemy blockhouse would be placed there or that the journey to that point would reveal the location of the blockhouse. If the diagonally opposite point was PLAIN, a search was made for a nearby PLAIN point which was used as the important point.

6 The Competition

Since there was only one competitor, the competition did not take the form of six games on three different terrains. Rather, a number of competitions were run on many different terrains for the purposes of observing behavior. In eight competitions against the internal competitor on various terrains, Rommel earned four strategic victories, two tactical victories, one tactical defeat and one strategic defeat.

The first competition used a very complicated terrain. Neither Rommel nor the IC were able to locate the other's blockhouse. Rommel's tanks suffered less damage, and thus Rommel earned a tactical victory. (See Figure 1)

The second competition used a simpler terrain, but one which exposed a weakness of Rommel: because of intervening water, Rommel seemed unable to navigate from certain mountains to certain other mountains in this terrain. Thus, it cycled through important points, and ended up looping. In this competition, neither competitor was able to locate the opponent's blockhouse. Rommel suffered less tank damage, and was awarded a tactical victory. (See Figure 2)

The third and fourth competitions showcased the strengths of Rommel. The third used a very simple terrain, consisting mainly of three long rectangles of forest. The fourth used a
very complicated terrain. In both cases, Rommel's wide-open 'Big Rectangle' searching technique allowed it to spot the IC's blockhouse quickly. Also in both cases, Rommel's attacking tanks stopped at the edge of a forest, allowing it to take pot-shots at the IC's blockhouse while the IC was powerless to respond, since it could not see Rommel's tanks at the edge of the forest (the IC did not use Smoke information. [See Appendix 8.2]) The third and fourth competitions both resulted in strategic victories for Rommel. (See Figures 3, 4, 7 and 8)

The fifth and sixth competitions both used the same terrain, with the competitors switching sides. The terrain consists largely of a number of small forest patches between the opposing blockhouses. The fifth competition proceeded much as the third and fourth, with Rommel quickly locating and destroying the IC's blockhouse. (See Figure 5)

The sixth competition, however, indicated a weakness in Rommel and a strength of the IC. Rommel's movement ignored forests, and reduced the number of important points via 'Big Rectangles.' This resulted in movements on a large scale. The IC uses a larger number of important points, and tends to make short "hops". Since the terrain consisted of a number of small forest patches, this meant that the IC's tanks were most often moving along the edges of a forest, while Rommel's tanks were moving in the open. Rommel did indeed spot the IC's blockhouse quickly and even scored a hit on it, but its tanks moving in the open were ambushed by the IC's well-hidden tanks. The IC was never able to locate Rommel's blockhouse, but gave Rommel a tactical defeat by destroying or disabling all of Rommel's tanks. (See figures 5 and 9)

The seventh and eighth competitions both used the same terrain, with the competitors switching sides. The terrain was fairly simple, consisting of large banks of mountains. In the seventh competition, Rommel quickly located the IC's blockhouse, and sent all of its tanks after it. The IC's three defending tanks did a great deal of damage to Rommel's tanks, but Rommel was able to get three hits on the blockhouse and scored a strategic victory. (See figures 6, 10 and 11)

In the eighth competition, however, Rommel was not so fortunate. It did, as usual, find the IC's blockhouse quickly, but the IC's three defense tanks were able to defend its blockhouse. All of Rommel's tanks were disabled in the attempt. Eventually, a single IC tank was able to locate and destroy Rommel's blockhouse despite the presence of a single defense tank (which had collided with the blockhouse and lost its mobility.) This gave Rommel its only strategic defeat.
Conclusions

Rommel's navigation was definitely superior to the IC's. The IC made better use of local terrain, but was poor at planning and executing long journeys. Rommel was capable of planning routes at a higher, more abstract level, considering the global terrain more than the local. This made Rommel superior in locating the opponent's blockhouse. Rommel did not make good use of local terrain in navigation, but this flaw only hindered it in one of the eight competitions. Rommel did make good use of local terrain in attack, hiding in the forests. This feature gave Rommel two strategic victories.

Benefits of the Project

While the number of competitors in the competition is disappointing, the project has nonetheless produced some positive results.

Stimulating industrial and academic interest in multiple agent behavioral modelling was a stated goal of GISMO. Over 300 parties received descriptions of the GISMO environment, and 65 received the software.

The GISMO scenario has proven itself to be a truly worthwhile environment in which to examine and test multiple agent behavioral modelling algorithms. Many of the parties who received GISMO information have stated that they are using it and will continue to use it in this fashion.

Many of the features of the Rommel and the IC may be directly applicable to simulation systems such as IST's SAFOR testbed. The navigation algorithms of the two programs are particularly interesting. Both utilize the concept of important points, though their determination of important points is quite different. The IC chooses important points based on a lower level criterion: simply how to navigate about the terrain. Rommel chooses important points based on a higher level criterion: how to most efficiently and quickly search the entire terrain. The criteria for choosing points, and the algorithms for plotting paths may both be applicable to SAFOR.
8.1 Figures

The following figures describe the GISMO tournament. They are snapshots of actual GISMO battles. In each figure, Mountain terrain appears as black, Forest terrain as dark gray, and Water terrain as light gray. Plain terrain appears white.
Figure 1.

This is the terrain used for the first competition. Rommel's tanks are at the lower left, the IC's at the upper right. Note that this terrain has no Water elements.
Figure 2.

This is the terrain used for the second competition. Rommel's tanks are at the upper right, the IC's at the lower left.
Figure 3.

This is the terrain used for the third competition. Rommel’s tanks are at the left, the IC’s at the right. Note that this terrain has no Water elements.
Figure 4.

This is the terrain used for the fourth competition. Rommel’s tanks are at the lower center, the IC’s at the upper right.
This is the terrain used for the sixth competition. Rommel's tanks are at the right edge, the IC's are just left of center. This terrain was also used for the fifth competition, with the positions of the competitors reversed.
Figure 6.

This is the terrain used for the seventh competition. Rommel's tanks are at the lower left, the IC's at the upper right. This terrain was also used for the eighth competition, with the positions of the competitors reversed. Note that this terrain has no Water elements.
Figure 7.

This is a portion of the terrain at the end of the third competition. Rommel’s tanks 3 and 4 are poised on the edge of the forest, where they can fire on the IC’s blockhouse (which is beside tank B.) The IC’s tanks B, C and D are powerless to stop them, because a motionless tank on the edge of a forest can’t be seen, and the IC doesn’t use Smoke information produced when the tanks fire their weapons.
This is a portion of the terrain at the end of the fourth competition. The situation is much the same as at the end of the third competition, with Rommel’s tanks F, G and D (D is adjacent to F) shelling the IC’s blockhouse (amid tanks 1, 2 and 3) unmolested.
This is a portion of the terrain at the end of the sixth competition. The IC's tanks 1, 4 and 6 are well hidden in the corners of forest terrain, while Rommel's tanks B, C, D, F, G and H are exposed. At this point, all of Rommel's tanks had been destroyed or immobilized.
Figure 10.

This is a portion of the terrain at the end of the seventh competition. Rommel has managed to destroy the IC's blockhouse (which appears as tank A), but has used tanks 3, 4, 5, 6 and 7 to do it, most of which have been destroyed or immobilized.
This is a portion of the terrain at the end of the eighth competition. Rommel's tanks B, C and D have been destroyed attacking the IC's blockhouse (B is adjacent to D), and tank E, G and H have been destroyed by IC tank 6 en route. Tank F alone is not enough to protect Rommel's blockhouse from the IC's tank 5.
8.2 Game Description

The document in this appendix is a description of the GISMO game scenario, exactly as it was sent to potential competitors.
GISMO
GAME FOR INTELLIGENT SIMULATED MILITARY OPPONENTS

Game Description

Abstract:
GISMO is the Game for Intelligent Simulated Military Opponents. The goals of the project are to stimulate interest in Intelligent Simulated Forces, to collect and examine some ISF's, and to create a simple testbed in which ISF's may be compared to one another.

The goals are to be achieved by the creation of a game. Competitors are invited to write programs to 'Play the Game.' The competing programs communicate with a simple simulator via 2400-baud modems on toll-free telephone numbers. The simulator resolves conflicts and determines winners.

This document describes the parameters of the game which has been designed for the GISMO competition. It describes the overall scenario, the battlefield and terrain, the tanks, the methods of communication between competitors and the simulator, and various important algorithms used by the simulator (such as line-of-sight and checksum calculation.)
The Game:
The game is a simple battlefield simulation. In any conflict, each competitor will have a blockhouse, and will control seven tanks. The object of the game is to locate and destroy the enemy blockhouse.

Competing programs will communicate with the simulator via modem. First, they will receive a description of the terrain and the location of their blockhouse. They will then have one minute to formulate a plan. Before a minute has elapsed, they must tell the Simulator where to place their tanks. Tanks may be placed only on Plain terrain within a 25-grid radius of each player's blockhouse. Misplaced tanks will be placed 'near' their blockhouse by an arbitrary algorithm. Following tank placement, the competing programs will receive a status message from and give orders to their tanks once every two seconds until the game ends.

There are four ways for the game to end:

1) One competitor destroys the opponent's blockhouse
   Winner awarded Strategic Victory
   Loser given Strategic Defeat

2) One competitor destroys all opposing tanks
   Winner awarded Strategic Victory
   Loser given Strategic Defeat

3) One competitor surrenders
   Winner awarded Strategic Victory
   Loser given Tactical Defeat

4) Judges halt conflict (in case neither competitor has a chance of achieving victory conditions)
   Tactical Victory/Tactical Defeat or Draw assigned by Tiebreaker

The Tiebreaker is a numerical value calculated as follows: Score 10 points for each shell that hits the Blockhouse, and 1 point for each hit of damage sustained on each tank (including collisions & immobility due to water). The player with the lower score is awarded a Tactical Victory, and the opponent is given a Tactical Defeat. If the scores are equal, the game is ruled a Draw.
The Battlefield:
The battlefield will be a 256x256 cartesian grid, with (0,0) being the southwest corner and (255,0) being the southeast corner. At each point on the grid there will be an object or terrain. Objects can only be Friendly Tanks, Friendly Blockhouses, Enemy Tanks, or an Enemy Blockhouses. Terrain is limited to: Plain, Forest, Water, Mountain. Tanks cannot move off the edge of the battlefield, but they will not be damaged for trying. The effect of terrain on tanks is as follows:

```
<table>
<thead>
<tr>
<th>Tank Can:</th>
<th>Move Into</th>
<th>See Thru</th>
<th>Fire Thru</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Forest</td>
<td>Yes*</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Water</td>
<td>No**</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mountain</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
```

* Tanks cannot move at full speed through Forest.
** Tanks can move into water, but once there are immobile.

There is a special "non-object" called Smoke. It appears whenever a tank fires or is hit by a shell, and lasts for only one time impulse. It does not impede movement, vision or firing, and may appear at the same grid coordinates as another object. It simply serves as a visual confirmation of a tank firing or being hit. It can be seen over Water and Forest terrain, and other objects, but cannot be seen over Mountain terrain.

The Blockhouses:
Each competitor has one blockhouse. It cannot move and has no firepower. It does, however, report its status and what it can see just like a tank. It will be destroyed after being hit by three shells (at any distance). A blockhouse is not damaged when a tank collides with it (although the tank is damaged). When a blockhouse is destroyed, the game ends and the opponent is declared the winner.
The Tanks:

On each 2-second time impulse, the competitors will receive a tank status report. This message will tell the position, heading and overall status of each tank. Competitors will also receive an object report describing the type, position and status of visible objects. (See the appendix for the format of the reports received by competitors from the simulator.)

Within two seconds of receiving these reports, competitors give orders for their tanks to the simulator. These orders have two parts: a Movement part consisting of a direction and speed, and a Weapons part consisting of a turret direction, whether or not to fire the weapon, and where to fire. Note that there is no "Scan" order, since this information is relayed once per impulse automatically. All of these actions would theoretically happen simultaneously, but to avoid confusion tanks obey orders in a fixed order: first firing, then changing speeds, then moving the tank, and finally turning the turret. All firing of all tanks during a time impulse occurs simultaneously. Damage is calculated after firing but before changing speeds.

First: Fire
Then: Change Speeds
Then: Move
Then: Turn Turret

Tank Movement:
The simulated tanks have four speeds: Ahead Full, Ahead Half, Halted, Back Half. On any impulse, the tanks may accelerate or decelerate by only one speed (e.g. if moving Ahead Full, a tank could decelerate to Ahead Half, but could not Halt.) A tank cannot move at full speed while passing through Forest terrain, although it can move at full speed leaving and entering a forest. If a tank moves into a grid location with water, it immediately loses its mobility for the rest of the game, which is the equivalent of a hit of damage.

The tanks turn differently at different speeds (i.e. a tank moving Ahead Full cannot turn as sharply as a tank moving Ahead Half.) Thus, the result of a turn is based on the tank's speed as well as the direction it is currently facing. The tanks are limited to face in one of eight directions: N, NE, E, SE, S, SW, W, NW. The tanks' abilities to change direction at various speeds are summarized in tables to follow.
A moving tank can change direction by at most one compass point per time impulse. A Halted tank can change direction more quickly than a moving tank, turning up to two compass points per time impulse. A Halted tank facing N can turn to NE, E, NW or W. A Halted tank facing NE can turn to N, NW, E or SE. Of course, a Halted tank may also remain facing in the same direction.

A tank moving at half-speed moves one grid location per time impulse. The following table describes the Half-speed movement of a tank at #. In all cases, the tank's new heading is the same as its order.

<table>
<thead>
<tr>
<th>Heading</th>
<th>Speed</th>
<th>Order</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>+1/2</td>
<td>NW</td>
<td>7</td>
</tr>
<tr>
<td>N</td>
<td>+1/2</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>+1/2</td>
<td>NE</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>-1/2</td>
<td>NW</td>
<td>3</td>
</tr>
<tr>
<td>N</td>
<td>-1/2</td>
<td>N</td>
<td>4</td>
</tr>
<tr>
<td>N</td>
<td>-1/2</td>
<td>NE</td>
<td>5</td>
</tr>
<tr>
<td>NE</td>
<td>+1/2</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>NE</td>
<td>+1/2</td>
<td>NE</td>
<td>1</td>
</tr>
<tr>
<td>NE</td>
<td>+1/2</td>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>NE</td>
<td>-1/2</td>
<td>N</td>
<td>4</td>
</tr>
<tr>
<td>NE</td>
<td>-1/2</td>
<td>NE</td>
<td>5</td>
</tr>
<tr>
<td>NE</td>
<td>-1/2</td>
<td>E</td>
<td>6</td>
</tr>
</tbody>
</table>

A tank moving at Full speed moves two grid locations per time impulse - one in its original direction, and one in its new direction (which may be the same as the original direction.)

Consider a tank at # facing N

<table>
<thead>
<tr>
<th>Order</th>
<th>1st Move</th>
<th>2nd Move</th>
<th>Final Dir.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW</td>
<td>N to 1</td>
<td>NW to 2</td>
<td>NW</td>
</tr>
<tr>
<td>N</td>
<td>N to 1</td>
<td>N to 3</td>
<td>N</td>
</tr>
<tr>
<td>NE</td>
<td>N to 1</td>
<td>NE to 4</td>
<td>NE</td>
</tr>
</tbody>
</table>

Consider a tank at # facing NE

<table>
<thead>
<tr>
<th>Order</th>
<th>1st Move</th>
<th>2nd Move</th>
<th>Final Dir.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>NE to 1</td>
<td>N to 2</td>
<td>N</td>
</tr>
<tr>
<td>NE</td>
<td>NE to 1</td>
<td>NE to 3</td>
<td>NE</td>
</tr>
<tr>
<td>E</td>
<td>NE to 1</td>
<td>E to 4</td>
<td>E</td>
</tr>
</tbody>
</table>
Weapons:

Each tank is equipped with a large gun attached to a rotating turret. Like the tank itself, the turret can be pointing in one of eight directions: N, S, E, W, NW, NE, SE or SW. The turret may rotate independently of the tank. In one time impulse the tank can be ordered to fire and then move its turret. The turret can turn up to 90 degrees relative to the tank per time impulse (this is the same turning rate as a halted tank.) The turret will turn with the tank (e.g. if the tank is facing N, the turret is facing W and the tank turns to face NW, the turret will now face SW.) For each direction, there is a scope of grid locations which can be hit (see the Vision/Firing Scope Diagram.) Each tank has a limit of 40 rounds of ammunition. Once a tank runs out of ammunition, it can no longer fire its gun. If the tank has not fired in either of the previous two time impulses, then the tank's ammunition loader is primed and can reload the gun quickly, and thus the tank may fire again on the next time impulse; if it has fired in either of the two previous time impulses and has fired in this time impulse, then it must take a time impulse to reload, so it cannot fire in the next time impulse. (See the Tank Weapons State Diagram in the appendices.) The gun has an effective range of 100 grid squares. A shot will either hit or miss its target (there are no "partial" hits), determined randomly. The probability of hitting a target is a function of distance, how fast the shooter is moving, how fast the target is moving, and from which direction the target is hit (see the appendices for a complete description of the To-Hit Probability Function.) Within a 50-grid radius, a hit is considered catastrophic, and thus destroys the target tank. From 50 to 100 grids, two hits are required to destroy the target tank. (See Damage later in this document.) Note that three shells are always required to destroy a blockhouse, even at close range. If an attempt is made to fire on a location which is not within the scope of the gun's current direction, the fire command is considered invalid. Thus, it is ignored, no ammunition is used, and the gun can be fired in the next time impulse without waiting for reloading. This is not the case for firing at a target which is out of range or obstructed.

When not firing, the tank's weapon crew is 'Tracking' a target. They can track targets only in the direction of the scope of their gun, and to which they have a clear line-of-sight. Whether a particular tank is firing its gun or tracking a target, the target location is given by its x- and y-coordinates in the tank's slot in the outgoing Orders Packet.
Line-of-Sight:
Many aspects of the game are affected by whether or not there is a clear line of sight between two points. The algorithm for determining grid locations lying on the line of sight between two points is outlined in the appendices. The following chart indicates which activities are inhibited by objects or terrain in the line of sight, and what type of objects or terrain inhibit them.

<table>
<thead>
<tr>
<th>Action:</th>
<th>Blocked By:</th>
<th>Not Blocked By:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seeing Tanks</td>
<td>Forest, Mountain, Tanks, Blockhouses</td>
<td>Plain, Water</td>
</tr>
<tr>
<td>Seeing a Blockhouse</td>
<td>Forest, Mountain, Blockhouses</td>
<td>Plain, Water, Tanks</td>
</tr>
<tr>
<td>Seeing Smoke</td>
<td>Mountain</td>
<td>Plain, Water, Forest, Tanks, Blockhouses</td>
</tr>
</tbody>
</table>

Note that a destroyed tank cannot see anything. Thus, this chart only applies to operational tanks.

Tanks have limited angles of visibility. They can see only within one "wedge" of scope in the direction the tank is facing, and three "wedges" of scope centered in the direction the tank's turret is facing. The wedges referred to here are identical to those illustrated in the Vision/Firing Scope Diagram later in this document. For example, if a tank is facing North and its turret is facing South, the tank could see North, South, Southeast and Southwest.

It should particularly be noted that Line-Of-Sight is defined as grid locations between and not including two points. Thus, a tank on the edge of a forest has a clear Line-Of-Sight out of the forest.

A tank can fire at any tank or blockhouse if there is a clear line-of-sight between shooter and target. However, there is a special case when a tank may be within the line-of-sight of another tank but still cannot be seen. A tank on the edge of a forest may still not be seen even if another tank has a clear line-of-sight. If tank A is on the edge of a forest and tank B has a clear line-of-sight to tank A, then tank B still cannot see tank A if tank A is stationary, and can see tank A 50% of the time (chosen randomly) if tank A is moving. Note that Tank B can still fire at Tank A if there is a clear line-of-sight between them even if Tank A can't be seen.
Vision/Firing Scope Diagram:
The following diagram describes the scope of a turret aimed in a given direction. A letter in a square indicates that it can be hit by a gun facing in either of the border directions. For example, if the tank at # has its gun facing North, then it can hit any square labeled A or B, or anything in between. If its gun is facing Northeast, then it can hit anything labelled B or C, or anything in between. All the squares labelled B can be hit by the tank if its gun is facing North or Northeast.
The above diagram is summarized in the following table:

Let \((X_f, Y_f)\) be the location of the Firing tank and \((X_t, Y_t)\) be the location of the Target.

Let \(Dx = X_t - X_f\), \(Dy = Y_t - Y_f\). Then,

\[
\begin{array}{|c|c|}
\hline
\text{Gun Direction:} & \text{Can Hit If:} \\
\hline
N & Dy \geq -2Dx \ \text{And} \ \ D_y \geq 2Dx \\
NE & Dy \leq 2Dx \ \text{And} \ \ D_y \geq Dx \\
E & 2Dy \leq Dx \ \text{And} \ 2Dy \geq -Dx \\
SE & 2Dy \leq -Dx \ \text{And} \ D_y \geq -2Dx \\
S & Dy \leq -2Dx \ \text{And} \ D_y \leq 2Dx \\
SW & 2Dy \leq Dx \ \text{And} \ D_y \geq 2Dx \\
W & 2Dy \geq Dx \ \text{And} \ 2Dy \geq -Dx \\
NW & 2Dy \geq -Dx \ \text{And} \ D_y \leq -2Dx \\
\hline
\end{array}
\]

Collision Resolution:

No two objects can occupy the same grid location (with the exception of the 'non-object' smoke). If a tank attempts to move into a grid location with another object or impassable terrain, or two tanks attempt to move into the same grid location, then a collision occurs.

If a tank is moving at half speed, then in any collision the following happens: the tank remains in its old location, but faces in the direction it attempted to move and permanently loses its mobility. This is the result if the tank attempts to move onto a mountain (forest and water are treated separately, however.) If a tank attempts to move into a grid location with a stationary tank, then both tanks lose their mobility for the rest of the game. If two (or more) tanks attempt to move into the same grid location, then all of the tanks collide and suffer mobility damage.

A full-speed move is equivalent to two half-speed moves. Therefore, collision resolution must occur in two phases: First, Half-speed and the first move of full-speed, and then the second move of full-speed (with all tanks not moving in a given phase considered stationary for that phase.)

The effect of one collision may cause another. Consider the following example:

<table>
<thead>
<tr>
<th>C</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

If all three tanks are moving East and are ordered to continue moving straight, no collision occurs. Tank A moves into the space vacated by Tank B. However, if tank C is moving South, it will collide with B at 1. Since B doesn't move, it will then collide with A which is ordered into its location. Many tanks may be involved in such a collision.
Damage:

Tanks incur damage when they are hit by fire, or when they collide with other tanks, blockhouses or impassable terrain. They do not incur damage if they try to move off of the battlefield. Each tank can take two hits. The first hit causes the tank to lose all moving capability. The second hit causes the loss of weapons capability, rendering the tank destroyed. Once a tank is destroyed, it no longer reports objects within its line-of-sight, although it still reports its position. A shell fired from within a 50-grid radius does 2 hits of damage to a tank, thus immediately destroying the target tank if successful. A shell fired from 50 to 100 grid squares does one hit of damage to a tank. A collision will destroy a tank's mobility, but not its weapon. Shells do the same amount of damage to a blockhouse regardless of the distance from which they were fired. Three shells are required to destroy a blockhouse under all circumstances.

It is possible for tanks (and blockhouses) to be hit by more than one shell during a single time impulse. When this happens, damage is computed in the usual manner, as though two successive hits had occurred. For example, suppose three of Player A's tanks simultaneously fire shells at one of Player B's undamaged tanks, and the distance between all of Player A's tanks and the target Player B tank is between 50 and 100 grids. The simulator determines (using the "To Hit" Probability Function) that two of the shells successfully hit the target, but one missed. The Player B tank would be destroyed, because the first shell would damage its mobility and the second one would damage its weapon.

Communications:

The simulator and the competitors communicate in units of information called packets. There are only a few kinds of packets, all of which are extensively documented in the appendices. Each packet begins with the string GISMO for alignment, and then each packet has a three-byte Protocol Block. The Protocol Block is a packet header which contains an ID of the packet, the length of the packet, and a checksum. Each packet sent by the simulator to competitors has a checksum which the competitors may use to verify the correctness of the packet. The checksum algorithm is given in the appendices. The competitors may or may not, at their option, include a checksum in the packets they send to the simulator. A Competitor never sends packets to the other competitor.

Before any communication can take place, the modems must establish connections. Once this has happened, the simulator will wait for the string GISMO to be sent by each competitor. This is for alignment purposes.
Once alignment has been established with both competitors, the simulator will send each competitor a Startup/Terrain Packet. This packet describes the battlefield terrain. The competitor should calculate the checksum of the Startup/Terrain Packet it has just received. (This is not mandatory, but it is strongly recommended.) If it matches the checksum in the Protocol Block, then the competitor should send a zero back in the Confirmation field of the Terrain Acknowledge Packet. If the checksums don't match, then the competitor should send back a non-zero value in the Confirmation field. After sending a Startup/Terrain Packet, the simulator will wait for a Terrain Acknowledge Packet from each competitor, looking for a success message. The simulator will try up to three times to send (and resend) the Startup/Terrain Packet to a competitor until it receives a success message in the subsequent Terrain Acknowledge Packet. If it fails three times, the simulation is aborted.

Once the simulator has received a correct Terrain Acknowledge Packet from both competitors, it will simultaneously send a Blockhouse Location Packet to each competitor. This tells the competitors the location on the battlefield of their blockhouse, and begins the one minute planning period.

Within one minute of sending a Blockhouse Location Packet to a competitor, the simulator will expect an Orders Packet from that competitor. The Firing Mask field of this packet is ignored, and the $i^{th}$ FireAtX and FireAtY fields are interpreted as the initial $(x,y)$ coordinates of the $i^{th}$ tank. The $i^{th}$ Heading, Speed and Turret fields are interpreted as the initial heading, speed, and turret direction of the $i^{th}$ tank. Tanks must be placed within a 25-grid radius of the player's blockhouse, on any terrain other than Plain. No tank can be placed at the same location as previously placed tanks or at the same location as the blockhouse. If a tank placement is faulty, the simulator will place the tank 'near' its blockhouse by an arbitrary algorithm.

From this point on, the simulation proceeds in two-second time impulses. The simulator will send a Tank Report Packet and an Object Report Packet to each competitor. The Tank Report Packet contains information on the location, speed, heading, turret direction and health of the competitor's tanks and blockhouse. It also contains information about the status of the game (e.g. if the game is over, if there are technical difficulties, etc.)
The Object Report Packet contains information about the objects that the competitor's tanks can see, what kinds of objects they are and how they are moving, and which tanks can and cannot see them. Note that a destroyed tank will not report objects in its line of sight, but it will still appear in the Tank Report Packet.

Within two seconds of having sent both report packets to a competitor, the simulator expects either an Orders Packet or a Surrender Packet from the competitor. Then, after resolving the move, a Tank Report Packet and an Object Report Packet are again sent to the competitor, in that order. This sequence continues at two-second intervals until the game is ended. If the game ends, this fact and how it ended are contained in the Tank Report Packet.

During the game, there are several ways that communications can become disrupted. In most situations when the simulator detects a bad stream of data from one of the competitors, it will simply ignore the data. Bad streams of data include missing GISMO alignment strings, incorrect checksums in the packets, or invalid orders in an otherwise correctly-formed packet.

When the simulator finds no GISMO string where it expects the beginning of a packet from the competitor, it will go into a loop waiting for GISMO. The simulator will count up the number of bytes that are received before finding the beginning of a GISMO string. If this number reaches 100 for a given packet, the game will be aborted due to "Technical Difficulties". Note that this counter is reset to zero every time GISMO is successfully found. The game will also be aborted if the number of instances of finding spurious data before a GISMO string reaches 3.

When the simulator receives a packet which has a nonzero value in the checksum field of the protocol block, it will compute the checksum of the data after the protocol block in the packet to make sure that it matches the transmitted checksum. See the appendices for the algorithm to compute checksums of packets. Note that this algorithm never produces zero as the value of a checksum, so that zero can be used as a flag for "ignore checksum". Each time a packet is received with a mismatch between the transmitted checksum and the computed checksum, a counter is incremented. If this counter reaches 3, the game is aborted due to "Technical Difficulties".

The information in this section is summarized in the Communications Protocol section of the appendices.
"To Hit" Probability Function:

If the distance \( D \) between the shooter and the target is greater than 100 grid squares, then the probability of hitting is automatically zero. If not, it is calculated in the following way: Let \((X_f,Y_f)\) be the location of the firing tank, and \((X_t,Y_t)\) be the location of the target. Let \(D_x = X_t - X_f\), \(D_y = Y_t - Y_f\), \(D^2 = D_x^2 + D_y^2\) (\(D^2\) is the square of the distance between the source and the target.) Then the base probability (expressed as a percentage) of hitting is

\[
P = \text{Round}( 100 \times \exp(-3 \times 10^{-4} \times D^2) - 5 )
\]

This base probability is then adjusted by adding the following modifiers:

- Speed of Tank: Modifier: Speed of Target: Modifier:
  -0.5 -3 -0.5 -5
  0.0 0 0.0 0
  0.5 -3 0.5 -5
  1.0 -5 1.0 -10

- If tank changed speed this turn: -5
- If tank changed direction this turn: -5
- If target changed speed this turn: -5
- If target changed direction this turn: -5

Direction of Impact: (only applies if target is a tank.) Tanks are more heavily armored in front. Thus, the direction from which a shell hits affects the probability that the hit will do significant damage. For example, a head-on shot incurs a -5 point penalty, while a rear-end shot gets a +5 point bonus.) Let \# be the target tank, and let 0 represent the direction the tank is facing:

\[
\begin{array}{cccccccc}
7 & 0 & 1 & \text{\textbackslash/\textbackslash} & \text{\textbackslash/\textbackslash} & \text{\textbackslash/\textbackslash} & 6 & 7 & 0 \\
6-\#-2 & 5-\#-1 & \text{\textbackslash/\textbackslash} & \text{\textbackslash/\textbackslash} & 5 & 4 & 3 \\
\hline
5 & 4 & 3 & 4 & 5 & 6 & 7 & 0 & -5
\end{array}
\]

Shot from: 0 1 2 3 4 5 6 7

If target was tracked or fired at last turn: +10

If the adjusted \( P > 95 \), then \( P = 95 \). (The probability of hitting can never be greater than 95\%)
Chart of Hit Probability as a Function of Distance:

The following chart shows the base probability $P$ of hitting a target (expressed in percent) as a function of distance $D$ (expressed in grids). This chart is merely a table of values computed using the formula for $P$ provided on the previous page. Note that this chart does not include the effects of any of the probability modifiers.

<table>
<thead>
<tr>
<th>D</th>
<th>P</th>
<th>D</th>
<th>P</th>
<th>D</th>
<th>P</th>
<th>D</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>95</td>
<td>26</td>
<td>77</td>
<td>51</td>
<td>41</td>
<td>76</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>95</td>
<td>27</td>
<td>75</td>
<td>52</td>
<td>39</td>
<td>77</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>95</td>
<td>28</td>
<td>74</td>
<td>53</td>
<td>38</td>
<td>78</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>95</td>
<td>29</td>
<td>73</td>
<td>54</td>
<td>37</td>
<td>79</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>94</td>
<td>30</td>
<td>71</td>
<td>55</td>
<td>35</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>94</td>
<td>31</td>
<td>70</td>
<td>56</td>
<td>34</td>
<td>81</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>94</td>
<td>32</td>
<td>69</td>
<td>57</td>
<td>33</td>
<td>82</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>93</td>
<td>33</td>
<td>67</td>
<td>58</td>
<td>31</td>
<td>83</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
<td>34</td>
<td>66</td>
<td>59</td>
<td>30</td>
<td>84</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>92</td>
<td>35</td>
<td>64</td>
<td>60</td>
<td>29</td>
<td>85</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>91</td>
<td>36</td>
<td>63</td>
<td>61</td>
<td>28</td>
<td>86</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>91</td>
<td>37</td>
<td>61</td>
<td>62</td>
<td>27</td>
<td>87</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>90</td>
<td>38</td>
<td>60</td>
<td>63</td>
<td>25</td>
<td>88</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>89</td>
<td>39</td>
<td>58</td>
<td>64</td>
<td>24</td>
<td>89</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>88</td>
<td>40</td>
<td>57</td>
<td>65</td>
<td>23</td>
<td>90</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>88</td>
<td>41</td>
<td>55</td>
<td>66</td>
<td>22</td>
<td>91</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>87</td>
<td>42</td>
<td>54</td>
<td>67</td>
<td>21</td>
<td>92</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>86</td>
<td>43</td>
<td>52</td>
<td>68</td>
<td>20</td>
<td>93</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>85</td>
<td>44</td>
<td>51</td>
<td>69</td>
<td>19</td>
<td>94</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>84</td>
<td>45</td>
<td>49</td>
<td>70</td>
<td>18</td>
<td>95</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>83</td>
<td>46</td>
<td>48</td>
<td>71</td>
<td>17</td>
<td>96</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>81</td>
<td>47</td>
<td>47</td>
<td>72</td>
<td>16</td>
<td>97</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>80</td>
<td>48</td>
<td>45</td>
<td>73</td>
<td>15</td>
<td>98</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>79</td>
<td>49</td>
<td>44</td>
<td>74</td>
<td>14</td>
<td>99</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>78</td>
<td>50</td>
<td>42</td>
<td>75</td>
<td>13</td>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>
Line-Of-Sight Algorithm:
The following algorithm is used to determine the line-of-
sight between points \((X_0, Y_0)\) and \((X_1, Y_1)\).

Let \(Dx = X_1 - X_0, Dy = Y_1 - Y_0\).

The algorithm varies if \(|Dx| \geq |Dy|\) or not, if \(Dx \geq 0\) or not, and if \(Dy \geq 0\) or not. Thus, there are eight variations. Only the variation in which \(|Dx| \geq |Dy|\), \(Dx\geq0, Dy\geq0\) will be given.

\[x = 1, \quad y = 0, \quad \text{Blocked} = \text{False}\]

While \(x < Dx\ AND NOT\ \text{Blocked}\)
\[\text{If } (y+1) \times Dx \leq x \times Dy + Dx \ \text{DIV} \ 2 \text{ Then }\]
\[y = y + 1\]
\[\text{End If}\]
\[\text{If Battlefield}[X_0+x, Y_0+y] \text{ is a Blocker Then}\]
\[\text{Blocked} = \text{True}\]
\[\text{End If}\]
\[x = x + 1\]
\[\text{End While}\]

If \(|Dx|<|Dy|\) then the roles of \(x\) and \(y\) in the above algorithm are reversed. If \(Dx<0\) (or \(Dy<0\)) then \(x\) (or \(y\)) is decremented rather than incremented, and appropriate changes must be made to the While and If conditions. It should be noted that all values are integers, and that \(Dx \ \text{DIV} 2\) represents an integer division.

Checksum Algorithm:

Check sums are calculated for the data part of a packet only. Protocol Blocks are not figured into the checksum. Checksums are calculated with the following algorithm:

\[\text{Byte Checksum} = 0\]
For each data byte of message, in order, Do
\[\text{Flip All 8 Bits of Checksum (Bitwise NOT)}\]
\[\text{Rotate Checksum Left 1 Bit (MSB becomes LSB)}\]
\[\text{XOR Checksum with Next Data Byte}\]
End For
\[\text{If Checksum} == 0 \text{ Then Checksum} = 255\]

Note that this checksum algorithm is sensitive to the order of the data bytes. Thus, the checksum must be calculated in order from the first data byte of a packet to the last. Also, this algorithm will never produce a zero value for the checksum. This is because the GISMO protocol uses zero a flag to signify that the checksum field is to be ignored.
Communications Parameters:
GISMO's communications will occur along 2400-baud modems. Speeds greater than or less than 2400 baud will not be supported. Competing programs should use No Parity, 8 Data Bits, 1 Stop Bit.

Packets:
A description of the packets follows. Note that for alignment each packet must be preceded by the string GISMO. Part of the pertinent information is summarized in the table below - note that the Length is in bytes, and does not include the 5-byte GISMO or the 3-byte Protocol Block.

<table>
<thead>
<tr>
<th>Packet:</th>
<th>Length:</th>
<th>Packet Type ID:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orders</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>Tank Report</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>Object Report</td>
<td>$1 + 5N$</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$N = \text{number of objects (max 24)}$</td>
<td></td>
</tr>
<tr>
<td>Startup/Terrain</td>
<td>$1 + 5N$</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$N = \text{number of terrain elements (max 255)}$</td>
<td></td>
</tr>
<tr>
<td>Blockhouse Location</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Terrain Acknowledge</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Surrender</td>
<td>9</td>
<td>7</td>
</tr>
</tbody>
</table>
Communications Protocol

The following pseudocode provides a high-level description of the GISMO Simulator's communications behavior with respect to an external competitor.

wait for "GISMO" string from competitor;
repeat
    transmit Terrain/Startup packet;
    receive Terrain Acknowledge packet;
until Terrain Acknowledge confirmation = 0; /* give up after 3 tries */
transmit Blockhouse Location packet;
receive Orders Packet (initial tank placement); /* max wait = 60 seconds */
loop /* each time around this loop is another time impulse */
    transmit Tank Report packet;
    transmit Object Report packet;
    if the game is over then stop;
    receive Orders packet or Surrender packet; /* max wait = 2 seconds */
    update the simulation based on the competitors' orders;
end loop;

Tank Weapons State Diagram

The following state diagram is a representation of the behavior of the tanks' firing capabilities. All tanks start in state Q0.

- Each state transition occurs over the span of one time impulse.

A tank cannot fire when in state Q2.
### Protocol Block

Packet Length field does not include length of the **GISMO** or of the protocol block itself.

<table>
<thead>
<tr>
<th>Byte</th>
<th>High Bit</th>
<th>Low Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Orders

**Orders for Tank 1**

- Mask - Who's Firing?
- Heading | Speed | Turret Facing
- X - Coordinate to Fire at/Track
- Y - Coordinate to Fire at/Track

**Orders for Tank 7**

- Heading | Speed | Turret Facing
- X - Coordinate to Fire at/Track
- Y - Coordinate to Fire at/Track
### Tank Report

<table>
<thead>
<tr>
<th>Byte</th>
<th>High</th>
<th>Bit</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NO</td>
<td>GO</td>
<td>ST</td>
</tr>
<tr>
<td>4</td>
<td>Heading</td>
<td>Speed</td>
<td>Turret Facing</td>
</tr>
<tr>
<td>5</td>
<td>M</td>
<td>W</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>X - Coordinate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Y - Coordinate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Heading</td>
<td>Speed</td>
<td>Turret Facing</td>
</tr>
<tr>
<td>29</td>
<td>M</td>
<td>W</td>
<td>A</td>
</tr>
<tr>
<td>30</td>
<td>X - Coordinate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Y - Coordinate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Byte Descriptions:****

- **0**: Packet Type ID = 2
- **1**: Packet Length = 29
- **2**: Checksum
- **3**: NO, GO, ST, VD, DR, TD, Block
- **4**: Heading, Speed, Turret Facing
- **5**: M, W, A, C, S, R, MI, WI
- **6**: X - Coordinate
- **7**: Y - Coordinate
- **28**: Heading, Speed, Turret Facing
- **29**: M, W, A, C, S, R, MI, WI
- **30**: X - Coordinate
- **31**: Y - Coordinate

**Status Descriptions:**

- **NO**: I = No Orders Received
- **GO**: I = Game Over
- **ST**: I = Strategic, 0 = Tactical
- **VD**: I = Victory, 0 = Defeat
- **DR**: I = Draw
- **TD**: I = Technical Difficulties
- **Block**: Blockhouse Status
  - (Number of Hits)
- **M**: I = Tank is still mobile
- **W**: I = Tank's weapon is still operational
- **A**: I = Tank still has ammunition
- **C**: I = Collision occurred
- **S**: I = Tank was hit by a shell
- **R**: I = Tank's weapon is reloading
- **MI**: I = A movement order was ignored
- **WI**: I = A weapon order was ignored
```plaintext
<table>
<thead>
<tr>
<th>Byte</th>
<th>High</th>
<th>Bit</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>packet type ID = 3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>packet length = 5n + 1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>checksum</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>n = number of objects</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>obj ID</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>who saw it? (bit mask)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>x - coordinate</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>y - coordinate</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>heading</td>
<td>speed</td>
<td>turret facing</td>
</tr>
</tbody>
</table>

```

**Object Report**

**Report for Object 1**

**Report for Object n**

- **Obj ID:**
- **Bit Pattern:**
  - Enemy Tank: 01
  - Enemy Blockhouse: 10
  - Smoke: 11

F: 1 = This tank just fired its weapon
T: 1 = This tank/blockhouse is being tracked
M: 1 = This tank is mobile
W: 1 = This tank's weapon is operational
<table>
<thead>
<tr>
<th>Byte</th>
<th>High Bit</th>
<th>Low</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>Packet Type ID = 4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Packet Length = 5n + 1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>Checksum</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>n = Number of Terrain Elements</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>X - Coordinate of NW Corner</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>Y - Coordinate of NW Corner</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>X - Coordinate of SE Corner</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>Y - Coordinate of SE Corner</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>Terrain Type</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>Description of Terrain Element 1</td>
</tr>
<tr>
<td>5n-1</td>
<td></td>
<td></td>
<td>X - Coordinate of NW Corner</td>
</tr>
<tr>
<td>5n</td>
<td></td>
<td></td>
<td>Y - Coordinate of NW Corner</td>
</tr>
<tr>
<td>5n+1</td>
<td></td>
<td></td>
<td>X - Coordinate of SE Corner</td>
</tr>
<tr>
<td>5n+2</td>
<td></td>
<td></td>
<td>Y - Coordinate of SE Corner</td>
</tr>
<tr>
<td>5n+3</td>
<td></td>
<td></td>
<td>Terrain Type</td>
</tr>
</tbody>
</table>

Terrain Types: 1 = Forest
               2 = Water
               3 = Mountain
### Blockhouse Location

<table>
<thead>
<tr>
<th>Byte</th>
<th>High Bit</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Packet Type ID** = 5
- **Packet Length** = 2
- **Checksum**
- **X - Coordinate of Blockhouse**
- **Y - Coordinate of Blockhouse**

### Terrain Acknowledge

<table>
<thead>
<tr>
<th>Byte</th>
<th>High Bit</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Packet Type ID** = 6
- **Packet Length** = 1
- **Checksum**
- **Confirmation**

**Note:** Confirmation is Zero if the competitor has received the Startup/Terrain message correctly (or if the competitor does not wish to verify the Startup/Terrain message.) Confirmation is NON-Zero if the competitor has received a malformed Startup/Terrain message and needs the simulator to resend it.
Surrender

<table>
<thead>
<tr>
<th>Byte</th>
<th>High</th>
<th>Bit</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>'S'</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>'U'</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>'R'</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>'R'</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>'E'</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>'N'</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>'D'</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>'E'</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>'R'</td>
<td></td>
</tr>
</tbody>
</table>

Note: This packet includes the ASCII string 'SURRENDER' in order to minimize the probability of mistaking a non-surrender packet for a surrender packet.
<table>
<thead>
<tr>
<th>Heading:</th>
<th>Bit Pattern:</th>
<th>Speed:</th>
<th>Bit Pattern:</th>
<th>Turret Facing: (Relative to Tank)</th>
<th>Bit Pattern:</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>000</td>
<td>Ahead Full</td>
<td>10</td>
<td>Ahead</td>
<td>000</td>
</tr>
<tr>
<td>Northeast</td>
<td>001</td>
<td>Ahead Half</td>
<td>01</td>
<td>Ahead Right</td>
<td>001</td>
</tr>
<tr>
<td>East</td>
<td>010</td>
<td>Halted</td>
<td>00</td>
<td>Right</td>
<td>010</td>
</tr>
<tr>
<td>Southeast</td>
<td>011</td>
<td>Back Half</td>
<td>11</td>
<td>Back Right</td>
<td>011</td>
</tr>
<tr>
<td>South</td>
<td>100</td>
<td></td>
<td></td>
<td>Back</td>
<td>100</td>
</tr>
<tr>
<td>Southwest</td>
<td>101</td>
<td></td>
<td></td>
<td>Back Left</td>
<td>101</td>
</tr>
<tr>
<td>West</td>
<td>110</td>
<td></td>
<td></td>
<td>Left</td>
<td>110</td>
</tr>
<tr>
<td>Northwest</td>
<td>111</td>
<td></td>
<td></td>
<td>Ahead Left</td>
<td>111</td>
</tr>
</tbody>
</table>

In any mask, each bit represents a tank, with bit 0 representing the blockhouse and bits 1-7 representing tanks 1-7.

<table>
<thead>
<tr>
<th>Terrain Type:</th>
<th>Bit Pattern:</th>
<th>Object ID:</th>
<th>Bit Pattern:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forest</td>
<td>000000001</td>
<td>Enemy Tank</td>
<td>01</td>
</tr>
<tr>
<td>Mountain</td>
<td>00000010</td>
<td>Enemy Blockhouse</td>
<td>10</td>
</tr>
<tr>
<td>Water</td>
<td>00000011</td>
<td>Smoke</td>
<td>11</td>
</tr>
</tbody>
</table>
8.3 Rules of the Competition

The document in this appendix is a description of the overall rules of the competition, exactly as it was sent to potential competitors.
GISMO

GAME FOR INTELLIGENT SIMULATED MILITARY OPPONENTS

Rules

1. Any School, Company, Individual or Group of Individuals may enter, with the exception that students and employees affiliated with UCF or IST are ineligible.

2. Any hardware may be used.

3. The competition consists of writing a program to control a small force of tanks in a simple scenario. Programs communicate (via 2400-baud modems on toll-free telephone numbers) with a simple simulator at IST, and compete directly and interactively with each other.

4. A complete description of the game, simulator and communications parameters will be released after November 5, 1990. At any time after that date, a description of the game, simulator and communications parameters will be sent to competitors upon receipt at IST of an expression of interest by the competitors.

5. Competitors will have approximately one year to develop their programs. The competition will begin in October, 1991. Until that time, competitors will have access to the Game Simulator at IST for testing and practice bouts.

6. Competitors' programs must be completely self-reliant. Once connection to the IST Game Simulator has been established, no human intervention is allowed until the conflict is resolved.

7. A copy of all software and data used in the final program must be submitted in electronic form to IST before the contest finals begin. The software must be submitted in source code form. This includes all software and data used by the program, and any software which is not part of the final program but which was developed to be useful solely for developing the algorithms and data in the final program. This does not include general utilities developed in conjunction with the project which are not directly related to the nature of the game. The software may be proprietary, and IST guarantees that it will abide by any proprietary notices placed in the source code.

8. After programs and data have been submitted to IST, competitors may not modify them.
9. The style of the tournament (Round-Robin, Double Elimination, etc.) will be determined by the number of competitors. Initial pairings will be random.

10. Finals will be held at IST in late October and early November of 1991. Results will be announced and published.

11. A match will consist of six conflicts, with both competitors taking each side on each of three different terrains. Each conflict will be scored as follows:

   Points:
   
   4  Strategic Victory
   3  Tactical Victory
   2  Draw
   1  Tactical Defeat
   0  Strategic Defeat

12. If a competing program achieves victory conditions, it is awarded a Strategic Victory and its opponent is given a Strategic defeat.

13. If a competing program surrenders, it is given a Tactical Defeat and its opponent is awarded a Strategic Victory.

14. If a conflict or match is terminated due to a hardware failure at either competitors' site or at IST, the match will be rescheduled.

15. If a conflict is terminated due to a software failure in a competing program, the competitor experiencing the software failure is given a Strategic Defeat, and its opponent is awarded a Strategic Victory.

16. If it becomes apparent that neither competing program is capable of achieving victory conditions, the judges will terminate the conflict. Each competitor will be given a Tactical Victory, Tactical Defeat or a Draw depending on the strength of remaining forces.

17. In order for the rules to be properly enforced, it is desirable (but not always possible) for a representative of IST to be present with each competing program during a match. Arrangements will be made with each competitor on an individual basis at the appropriate time.
8.4 Documentation provided with GAME.EXE

The document in this appendix is a reprinting of a file which was provided along with the GISMO software to describe the GAME.EXE simulator, how to use it and how to compile it.
GAME.EXE - The GISMO Simulator

Introduction

The GISMO simulator is a program which implements the rules of GISMO in their entirety for use in debugging and testing computer programs that will compete in the GISMO competition. It is functionally equivalent to the simulator that will be used in the actual GISMO competition in October/November 1991.

The GISMO simulator is distributed on disk in two different forms: as an executable MS-DOS (3.0 or above) file (GAME.EXE) and as complete Turbo C source code. The program requires an IBM-PC, AT, PS/2, or compatible, with a VGA monitor for graphics. The machine should also have one or two serial ports, with at least one Hayes Smartmodem compatible modem attached to one of the serial ports. Because the simulator produces an ongoing log file of the game being played, we recommend that the machine also have a hard drive so that the simulator can maintain a good execution speed.

How to use GAME.EXE

To use the simulator, you need two files: GAME.EXE and BATTLE.FLD. The file BATTLE.FLD contains a description of the terrain and players' blockhouse locations in a sample battle scenario. You can use the Terrain Editor program (TED.EXE) to modify this file, or to create new .FLD files (see TED.DOC for explanation of the Terrain Editor and a description of the format of .FLD files).

After copying GAME.EXE and BATTLE.FLD onto your hard drive, you can run the simulator program by typing

GAME <enter>

at the DOS prompt. The program will ask you questions to determine how it is to be run. First it will ask you whether Player A is to be a Remote (R) or an Internal (I) player. If you choose Internal, the simulator will use an internal algorithm to play the role of Player A in the competition. (The internal competitor is not very smart. In fact, it frequently does very unintelligent things. We have included it merely as a convenience so that you can debug your competitor programs by having a "drone" opponent to play.) If you choose Remote, the simulator will ask you for a serial port number (1 through 4 for COM1 - COM4) to use to connect to the remote competitor. After answering the questions about Player A, the simulator will ask you the same questions about Player B. If you have specified at least one remote competitor, the simulator will then wait for a competitor to establish communications with it on the specified serial port(s). (See the GISMO printed documentation's description of the serial communication protocol.) At this time, you should run your competitor program(s) on the remote machines as specified.
There are two ways to attach a remote competitor to the GISMO simulator. The first way is to use a "null-modem" cable to directly connect the serial ports of the competitor and the simulator. A null-modem cable has the receive pin of one side of the cable attached to the transmit pin on the other side of the cable, and vice versa. This is so that data transmitted from one machine is immediately received by the other machine due to direct pin-to-pin electrical connection. Null-modem cables can be purchased at most computer supply stores.

The other way to connect the simulator with the remote competitor is to use a modem on each machine, and have the competitor dial the simulator over the telephone system. It is important to understand that the competitor must send commands over the serial port to control the modem if a modem is being used, but should refrain from doing this if a null-modem cable is being used. You will probably need a command-line option or other operator-specified option to tell the competitor whether it is accessing a modem or a direct serial connection.

The GISMO simulator will work identically whether the serial connection is by modem or by a direct null-modem cable connection. This is why, when you tell the simulator that it has a remote competitor, that you need only tell it which serial port to use, and not whether it is connected to a modem or a null-modem cable.

The simulator will work with zero, one, or two remote competitors, although we anticipated that most of the time you will want to use one remote competitor, i.e. the program you are developing. Sometimes, however, you may want to have two copies of your program compete against each other, and so you would specify two remote competitors on separate serial ports.

After running the remote competitors, you will see a confirmation message on the simulator's graphics display each time a remote competitor establishes communications with it. Then, after all remote competitors establish communications, the game will begin. The game proceeds in two-second time impulses, as specified in the GISMO game documentation. The two-second time impulse is measured from the time the simulator sends out the Object Report packet to the competitors until two seconds later. The time impulse does not include the time it takes the simulator to update the state of the game, including processing the competitor's orders and updating the graphics display. Therefore, a time impulse may take 2.5 seconds or more, depending on the speed of the host machine.
The GAME simulator graphics display is organized in a similar manner to that of TED, the Terrain Editor. There is a radar screen on the left, which shows an overall view of the battlefield, and there is a zoom window on the right, which magnifies a small section of the battlefield for closer viewing. In the radar screen, each tank is represented as a single pixel, colored cyan if it is a Player A tank/blockhouse, and red if it is a Player B tank/blockhouse. Forests, Water, and Mountains appear as rectangles colored green, blue, and light gray, respectively.

After each time impulse, the simulator writes a binary record to the file GAME.LOG. This file contains complete information about the state of the game during each time impulse. After the game is over, you can use the ViewGame utility to review exactly what happened during the game, including all the competitor orders, locations of the tanks, which tanks saw which other tanks, and how the competitor orders were interpreted by the simulator. See the file VIEWGAME.DOC for more information on the ViewGame utility.

While the game is running, you can control several aspects of the graphics display by using keyboard commands. These are summarized below.

0..7 - Center Zoom on Player A tank 0..7
       (0 is blockhouse)
A..H - Center Zoom on Player B tank A..H
       (A is blockhouse)
Up Arrow - Move up one grid square
Down Arrow - Move down one grid square
Right Arrow - Move right one grid square
Left Arrow - Move left one grid square
Ctrl-Right - Move right 1/2 zoom screen
Ctrl-Left - Move left 1/2 zoom screen
Ctrl-PgUp - Move up 1/2 zoom screen
Ctrl-PgDn - Move down 1/2 zoom screen
Home - Move to upper left corner of battlefield
End - Move to lower left corner of battlefield
PgUp - Move to upper right corner of battlefield
PgDn - Move to lower right corner of battlefield
S - Toggle sound effects on and off
Escape - Abort game and return to DOS
How to compile GAME source code

The simulator source code is designed for compilation with Turbo C version 2.0. If you are using Turbo C++, you will still be able to compile the simulator, but you will first need to use the Borland utility PRJCNVT to convert the GAME.PRJ file to Turbo C++ .PRJ file format. In either case, you will need to use the file GAME.PRJ as your project file and compile the program with the following options set, or the program will not work correctly:

- Memory Model = Large
- Alignment = Byte
- Floating Point = Emulation
- Default Character Type = Signed

These are all VERY important!

You will need the following files to compile the simulator:

game.c The main source file
behavior.c GISMO rules source code
terrain.c Terrain, graphics, and initialization code
opponent.c The internal competitor
messages.c Routines for sending/receiving GISMO packets
useful.c Miscellaneous useful routines
gamelog.c Routines for creating GAME.LOG
modem.c Routines for controlling Hayes Smartmodem
serial.obj Routines for asynchronous serial port I/O
egavga.obj EGAVGA.BGI graphics driver converted to .OBJ
8.5 Documentation provided with TED.EXE

The document in this appendix is a reprinting of a file which was provided along with the GISMO software to describe TED.EXE, the terrain editor.
TED - The Terrain Editor
Part of the GISMO family of software tools

Introduction
TED (Terrain EDitor) is a program provided for your use in designing and modifying GISMO terrain scenarios. It runs on any IBM-PC, PS/2 or 100% compatible running MS-DOS 3.0 or above. TED requires a VGA monitor for graphics.

Command-line syntax
The command line syntax for TED is as follows:

TED [filename[.FLD]]

The filename specified may contain a drive letter, subdirectory path, etc., but must have an extension of "FLD". If the extension is omitted, the extension "FLD" is assumed. If you do not specify a filename, TED assumes the filename "NEW.FLD". Files with the extension "FLD" are GISMO battlefield terrain files.

When you run TED, it will check to see if the specified terrain file exists. If so, it will read it into memory so that you can look at it and/or make changes to it. If the given terrain file does not exist, TED will start with a new, blank battlefield.

Screen layout of TED
The screen layout in TED consists of three basic parts: the "radar" screen, the zoom screen, and the status indicators. The radar screen is a large white square on the left side of the display. It contains a picture of the entire battlefield's terrain. Inside the radar screen is a square which shows the position of the zoom screen's view inside the radar screen. The zoom screen (the square region on the right side of the display) serves as a magnified view of a small portion of the radar screen, which the user can move around the battlefield at will. Finally, the status indicators are the miscellaneous pieces of text outside the radar screen and the zoom screen, including the current cursor location's x and y coordinates, along with the type of terrain at the cursor and the name of the current terrain file. The cursor mentioned here is the very small green square inside the zoom screen window. The cursor is used in many of the functions of TED for inspecting, adding, and deleting items of terrain.
Summary of TED Commands

Here is a summary of the commands you will need to use TED:

<table>
<thead>
<tr>
<th>Key</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>F10</td>
<td>Save current terrain file and quit to DOS</td>
</tr>
<tr>
<td>F2</td>
<td>Save current terrain file</td>
</tr>
<tr>
<td>Alt-X</td>
<td>Quit TED without saving changes to terrain file</td>
</tr>
<tr>
<td>Up</td>
<td>Move up (north) 10 grid locations movement</td>
</tr>
<tr>
<td>Down</td>
<td>Move down (south) 10 grid locations commands</td>
</tr>
<tr>
<td>Right</td>
<td>Move right (east) 10 grid locations</td>
</tr>
<tr>
<td>Left</td>
<td>Move left (west) 10 grid locations</td>
</tr>
<tr>
<td>PgUp</td>
<td>Move northeast 10 grid locations</td>
</tr>
<tr>
<td>PgDn</td>
<td>Move southeast 10 grid locations</td>
</tr>
<tr>
<td>End</td>
<td>Move southwest 10 grid locations</td>
</tr>
<tr>
<td>Home</td>
<td>Move northwest 10 grid locations</td>
</tr>
<tr>
<td>Shift-Up</td>
<td>Move up (north) one grid location</td>
</tr>
<tr>
<td>Shift-Down</td>
<td>Move down (south) one grid location</td>
</tr>
<tr>
<td>Shift-Right</td>
<td>Move right (east) one grid location</td>
</tr>
<tr>
<td>Shift-Left</td>
<td>Move left (west) one grid location</td>
</tr>
<tr>
<td>Shift-PgUp</td>
<td>Move northeast one grid location</td>
</tr>
<tr>
<td>Shift-PgDn</td>
<td>Move southeast one grid location</td>
</tr>
<tr>
<td>Shift-End</td>
<td>Move southwest one grid location</td>
</tr>
<tr>
<td>Shift-Home</td>
<td>Move northwest one grid location</td>
</tr>
</tbody>
</table>

(* Note: The Num-Lock key is often useful for alternating between moving 10 grid locations and moving 1 grid location.)

Terrain Editing Commands

<table>
<thead>
<tr>
<th>Key</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Place player A's blockhouse at cursor</td>
</tr>
<tr>
<td>B</td>
<td>Place player B's blockhouse at cursor</td>
</tr>
<tr>
<td>M</td>
<td>Add mountain terrain element at cursor</td>
</tr>
<tr>
<td>F</td>
<td>Add forest terrain element at cursor</td>
</tr>
<tr>
<td>W</td>
<td>Add water terrain element at cursor</td>
</tr>
<tr>
<td>Del</td>
<td>Delete terrain element at cursor</td>
</tr>
<tr>
<td>O</td>
<td>Toggle display of terrain element boundaries</td>
</tr>
</tbody>
</table>

When adding terrain elements using the M, F, or W keys, you will notice four small dots inside the radar screen which correspond to the corners of the rectangular terrain item you are about to add. Then, by moving the arrow keys (or shift-arrow keys) around, you can control exactly how wide and high you want to make the terrain element. If you change your mind about a terrain element you are entering, you can press Escape at any time to cancel the new terrain element. TED will not allow you to overlap terrain elements, nor will it allow you to place either of the competitors' blockhouses inside terrain occupied by forest, water, mountain, or the other blockhouse.
The O command exists for situations in which you wish to modify terrain objects which are made of multiple terrain elements. For example, it is possible to create a maze out of a set of long, thin mountain terrain objects connected together. In such a situation, it may become difficult to tell which part of the terrain belongs to which rectangular terrain element. In this case, the O command will display the rectangular border of each terrain element on the radar screen. That way, you can position the cursor inside a rectangular terrain element and delete it with the Del key, if you wish. Pressing the O key again will toggle the border display back off, resuming the normal display.

File format of the terrain (.FLD) file

Data files produced by TED are in a very simple format. They are simply ASCII text files which contain several lines of integers, as follows:

Line 1: ax ay (x,y coordinates of Player A's blockhouse)
Line 2: bx by (x,y coordinates of Player B's blockhouse)
Line 3: N (number of terrain elements in file)
Line 4: xa1 ya1 xb1 yb1 t1 (coords & type of terrain element #1)
Line 5: xa2 ya2 xb2 yb2 t2 (coords & type of terrain element #2) ...
Line 3+N: xaN yaN xbN ybN tN (coords & type of terrain element #N)
(end of file)

In each line of terrain elements (lines 4 through 3+N), the first two integers represent the x and y coordinates of the northwest (upper left) corner, and the second two integers represent the coordinates of the southeast (lower right) corner of the terrain element. The terrain element is assumed to fill all of the grid locations (x,y) such that xa <= x <= xb, and yb <= y <= ya.

The final integer on each of these lines represents the type of terrain, as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Forest</td>
</tr>
<tr>
<td>2</td>
<td>Water</td>
</tr>
<tr>
<td>3</td>
<td>Mountain</td>
</tr>
</tbody>
</table>
8.6 Documentation provided with VIEWGAME.EXE

The document in this appendix is a reprinting of a file which was provided along with the GISMO software to describe VIEWGAME.EXE, a facility to replay and diagnose GISMO log files.
ViewGame - GISMO game debugging utility

Introduction

ViewGame is a program which can interactively view the contents of a GAME.LOG file produced by the GISMO simulator (GAME.EXE). It was originally written to aid in debugging the simulator, but we have distributed it to GISMO competitors because it will be useful for debugging your competitor programs also.

ViewGame is distributed as an MS-DOS executable file (VIEWGAME.EXE). It requires an IBM-PC, AT, PS/2, or compatible machine with a VGA or EGA monitor for its graphics display. A VGA monitor will give a higher resolution display, but an EGA monitor will also work.

ViewGame will allow you to step through the game, time impulse by time impulse (forward or backward) and view the complete state of a GISMO game during each time impulse. You can inspect the orders each competitor gave, how the simulator interpreted the orders, where each tank was, which tanks could see which other tanks, which tanks fired at which other tanks, whether the shots were successful, etc.

How to use ViewGame

The command line syntax of ViewGame is as follows:

    VIEWGAME [[d:] [path] filename.ext] <Enter>

where filename defaults to GAME.LOG if not specified on the command line. GAME.LOG is the name of the file that GAME.EXE produces, so unless you rename its output file, you will not need to specify a filename on the command line with ViewGame. Note that, if you do specify a filename, you must explicitly give the filename's extension (.LOG is *not* assumed).
After you press <Enter>, the program will load terrain information from the .LOG file and show the terrain, along with the initial tank placements, in a graphics display. ViewGame is controlled by keyboard commands, as summarized below.

- **Enter** - Toggle between graphics display and text display
- **0..7** - Center on Player A tank 0..7 (0 is blockhouse)
- **A..H** - Center on Player B tank A..H (A is blockhouse)
- **Escape** - Exit to DOS
- **Left, Right, Up, Down Arrows** - Move around screen
- **PgUp** - Move forward in time by one time impulse
- **PgDn** - Move backward in time by one time impulse
- **Ctrl-PgUp** - Move to first time impulse of the game
- **Ctrl-PgDn** - Move to last time impulse of the game
- **Alt-A** - Toggle animation mode on and off
- `'+'** - Increase zoom window scale
- `'-'** - Decrease zoom window scale
- **Alt-P** - Select point of view
  (simulator, Player A, Player B)

Note that ViewGame provides two alternative ways of looking at the game: the graphics display and the text display. Each of these two facilities has its advantages for doing certain kinds of debugging work. The graphics display is better at getting an intuitive view of what happened in the game, while the text screen provides a numerical table of all the tanks' orders and status reports.

In the graphics display, whenever a tank/blockhouse can see another tank/blockhouse, you will see a cyan line connecting them, with a small circle around the tank doing the seeing, in the radar screen. When a tank is firing at another tank, you will see a red line with a small red circle around the tank which is doing the shooting.

In the zoom window of the graphics display, tanks and blockhouses are represented by cyan or red squares (depending on whether they belong to Player A or Player B, respectively) with special status markings rendered in white. Tank markings include a marking for the heading of the tank (a half rectangle with the solid edge pointing in the same direction as the tank's heading) and a marking for the tank's turret (a single line pointing in the same direction as the tank's turret). If a tank loses its mobility, its heading marking will disappear. Likewise, if a tank loses its weapons capability, its turret marking will disappear. Therefore, a completely destroyed tank will appear as a colored rectangle with no markings.

Blockhouses in the graphics display are very similar to tanks, except that they are depicted with an 'X' symbol rendered with two white lines. The blockhouse symbol in the zoom display always looks like this, regardless of how many hits of damage it has sustained. One must use the text display to see how much damage a blockhouse has taken.
The text display consists of four sections: the Status section, the Orders section, the "Seen By" section, and the Smoke section. Tanks/blockhouses are oriented vertically on the screen, and the first three of these sections are arranged horizontally on the screen.

The Status section gives you the Location, Heading, Speed, Turret direction, Damage level, Reload state, Ammunition count, Hit status and Collision status of each tank. Here is a description of each of these fields of the Status section in more detail:

**Loc:** Location of tank; (x,y) coordinates.

**Hd:** Tanks heading (0..7) where 0=North, 1=Northeast, 2=East, etc.

**Sp:** Tank's speed, where -1=reverse, 0=halted, 1=half speed ahead, 2=full speed ahead.

**Tr:** Turret heading RELATIVE TO tank heading. (Tr + Hd) & 7 gives you the actual turret absolute heading, as defined for Hd.

**Dmg:** Displays both a numerical value of the number of hits of damage the blockhouse/tank has sustained, along with the letters M and W. If an M (W) is green, it means the mobility (weapon) is still functional. If the M (W) is red, the mobility (weapon) has been destroyed.

**Re:** The reloading state is a numerical value that tells what state the tank's ammunition reloader is in. 0 means that the tank's loader is fully primed and is capable of firing two shots in a row for two time impulses. 1 means that the tank can fire a shot now, but must wait a time impulse afterwards to reload before firing again. 2 means the tank is currently reloading, and cannot fire a shot.

**Am:** The number of rounds of ammunition left in this tank. Starts out at 40. When it reaches 0, the tank is out of ammunition and cannot fire any more.

**Ht:** The 'Hit by Shell' flag. This flag will have a value of 0 except when the tank is successfully hit by a shell. Then, the Ht flag will take on a value of 1 for that time impulse.

**Co:** The 'Collision' flag. This flag will have a value of 0 except when the tank collides (or is collided by) another tank/blockhouse. Then, the Co flag will take on a value of 1 for that time impulse.
The next section of the display is the Orders section. This section contains a list of all the orders each competitor gave to each of their tanks. Below is a description of each of the fields in the Orders section:

**Target:** Shows the (x,y) coordinates of the location on the battlefield that the given tank is targeting. If this field is printed with a black background, the tank is only tracking the given location. However, if the background is printed red, the tank has been ordered to fire a shell at the given location. If a tank is currently at the given location, it is printed following a colon after the coordinates of the target. Important note: Because there are several phases in the simulator's computation of what happens during a time impulse, some of the things ViewGame displays may seem incorrect, even though they are actually just fine. An example of this is when a tank fires at another tank which is moving. In the graphics display and the text display, it will appear that the tank is firing at the location where the tank was in the previous time impulse, rather than where the tank currently is. This happens because the simulator processes all firing orders, then all movement orders. Then it saves the state of the game to the log file, including both the firing orders and the movement orders. To the user of the ViewGame program, it may appear that the tank is firing at the wrong location, so the user must be careful to remember what is really happening.

**Hd:** The ordered new heading of the tank.

**Tr:** The ordered new turret position of the tank, relative to its heading.

**WI:** A weapon order was ignored this time impulse.

**MI:** A movement order was ignored this time impulse.

The third section is the "Seen By" section. It shows which tanks/blockhouses were seen by which enemy tanks/blockhouses, as determined by the GISMO rules for tank visibility. These rules include both terrain which blocks vision (Forests and Mountains), as well as turret and heading vision scopes. (See the GISMO documentation for specifics.)

The final section of the ViewGame text display is the Smoke section. It displays a list of all instances of smoke, and a list of tanks/blockhouses which can see each of them. Finally, in the lower right hand corner of the screen is an indicator of which time impulse the user is viewing.
### 8.7 Acronym Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>Association for Computing Machinery</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Research Projects Agency</td>
</tr>
<tr>
<td>GISMO</td>
<td>Game for Intelligent Simulated Military Opponents</td>
</tr>
<tr>
<td>IC</td>
<td>Internal Competitor</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute for Electrical and Electronic Engineers</td>
</tr>
<tr>
<td>IST</td>
<td>Institute for Simulation and Training</td>
</tr>
<tr>
<td>SAFOR</td>
<td>Semi-Automated FORces (SAFOR is the name of IST's semi-automated forces testbed product)</td>
</tr>
<tr>
<td>SMU</td>
<td>Southern Methodist University</td>
</tr>
</tbody>
</table>