(1)

AFIT/GOR/ENS/88D-17



# A PROTOTYPE DECISION SUPPORT TOOL FOR ENGINEER COUNTERMOBILITY PLANNING

THESIS

Cynthia E. Staley
Captain, USA

AFIT/GOR/ENS/88D-17



Approved for public release; distribution unlimited

AFIT/GOR/ENS/88D-17

# A PROTOTYPE DECISION SUPPORT TOOL FOR ENGINEER COUNTERMOBILITY PLANNING

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Cynthia E. Staley, B.S.

Captain, USA

December 1988

## Acknowledgements

In conducting this research I am indebted to my thesis advisor, MAJ Bruce Morlan, and my reader, MAJ Dan Reyen. Their guidance and direction helped me to produce a product which balances Operations Research and Artificial Intelligence techniques with U.S. Army tactical doctrine. Also invaluable was Mr. Jeff Siferd in the Avionics Lab of the Aeronautical Systems Division who transcribed the digital terrain data needed for the model. Lastly, I would like to thank my Turbo Pascal guru the 'Raid Man', CAPT Christopher Baron, for his patience and debugging skills.

Cynthia E. Staley

## Table of Contents

## List of Figures

## List of Tables

This thesis proposes and demonstrates a method

AFIT/FOR/ENS/88D-17


## Abstract

The purpose of this research is to propose and demonstrate a methodology for determining the effect of combat engineer countermobility missions on an enemy's ability to manuever on the battlefield. The methodology combines implicit mobility modeling and existing digital terrain data to represent a movement area in the Federal Republic of Germany. An artificial intelligence technique, A* Search, is used to determine avenues of approach. The effects of obstacle emplacement on those approaches is quantified by both the time delay incurred and the effect on movement paths.

*The methodology is demonstrated in a computer based,* menu driven decision support tool written in Turbo Pascal. The prototype provides the user with graphical screens displaying elevation, terrain types, limiting speeds and avenues of approach in the study area. The user can select and place engineer obstacles on the avenues of approach and immediately see their effects on movement paths and times. The program can be used to evaluate different obstacle systems and channelization or delay plans.



Keywords: Countermobility; Army operations; Combat engineering; Tactical analysis; Airland battle doctrine; Computer programs; Theses, (edc)

# A PROTOTYPE DECISION SUPPORT TOOL FOR ENGINEER COUNTERMOBILITY PLANNING

## I. Introduction

### AirLand Battle Doctrine

AirLand Battle Doctrine is how the U.S. Army fights (9). AirLand Battle Doctrine is based on the principles of war and a desire to secure and retain the initiative necessary to defeat an enemy. The four basic tenets of AirLand Battle are: initiative, depth, agility, and synchronization (9:2-1).

Initiative refers to setting the terms of battle through decisive action. The commander sets the agenda by determining the time and place to fight. Whether U.S. forces are attacking or defending, commanders must seize any opportunity to hasten the enemy's defeat.

Depth acknowledges that the battlefield is nonlinear. Modern combat can extend throughout the entire theatre of operations and commanders must see beyond the immediate battle to balance rear, main, and deep area engagements.

Agility means acting and thinking faster than the enemy in order to exploit his weaknesses. Repeatedly interrupting the enemy's plan leads to ineffective and uncoordinated

responses and eventually results in the enemy being unable to execute his intended mission.

Synchronization combines economy of force and unity of effort. Economy of force pits the proper units against the enemy force. Unity of effort is the effective coordination of friendly forces. Synchronized, violent execution is the key to decisive combat.

Combat Engineers and AirLand Battle Doctrine

U.S. Army engineers support AirLand Battle doctrine in five primary areas: mobility, countermobility, survivability, general engineering and topography (7:1-10). These functional areas recognize that superior combat power derives from "artful combinations of manuever, firepower, protection, and intelligent leadership in a sound operational plan" (9:1-8).

Effective manuever depends on mobility to keep the force moving forward. Engineer mobility missions counter enemy minefields and obstacles, cross gaps, maintain and establish combat roads and trails and support forward aviation assets (7:1-10). In offensive operations, mobility is an engineer's first mission.

The ability to mass enemy forces into pre-established kill zones insures the effective use of firepower. Combat engineers accomplish this through countermobility missions. Countermobility tasks combine mine warfare and obstacle development to delay, disrupt, and channelize the enemy into

kill zones. Countermobility is the key to success in the defense.

Protection of personnel and equipment is the essence of combat engineer survivability tasks. These tasks include constructing protective fighting positions, weapon emplacements, and support facilities. Deceiving the enemy as to our purposes also serves to protect. To support deception, combat engineers can construct mock facilities and logistic areas.

General engineering and topographic missions provide support to the leadership and overall operation. General engineering tasks include: keeping lines of communication open, constructing logistics facilities, conducting area damage control and providing construction material production. Topographic support provides both terrain analysis and map production, reproduction and distribution.

The Importance of Countermobility Planning

As stated earlier, countermobility tasks are directly related to being successful in the defense, but the planning of countermobility missions is not easy.

First the terrain must be analyzed to determine how the enemy will most likely attack through the area. By eliminating terrain which cannot support the enemy's advance, avenues of approach are formed. These avenues of approach are the movement paths the enemy is expected to travel. Along the avenues of approach the engineer must select areas

where mine warfare and obstacle development will be particularly effective. Once the obstacle is selected, logistics, transportation, manpower, and indirect and direct fire assets must be coordinated. Obstacles do not stand alone: they must be integrated with the commander's plan to provide an effective barrier system.

Although it is a relatively simple process to determine the manpower and logistics to place an obstacle, it is not so easy to answer questions such as: Will the enemy choose to bypass or breach the obstacle? What effect will the decision to bypass or breach have on the enemy movement paths and passage times? The inability to answer these questions has led to "more is better" and "better safe than sorry" attitudes where as many obstacles as possible are placed leaving other engineer work areas with little or no planning and effort.

In the past, countermobility planning has concentrated on attrition and resource requirements rather than on the effects of delay and channelization. However, it can be argued that in a European scenario, attrition alone may not be sufficient to insure NATO forces accomplish their mission. Bingham discusses this concept relative to air interdiction.

> Because of the surprise and speed fundamental to Warsaw Pact doctrine, NATO's air interdiction objectives might better be served by focusing on delay and disruption rather than entirely on destruction (1:98).

Just as in air interdiction, ground interdiction is used to delay, disrupt, and to channelize the enemy and deny him important terrain. A methodology to better model and analyze delay and channelization is the focus of this thesis.

The main goals of this thesis are: (1) To propose a methodology for quantifying the effects of countermobility missions on enemy movement and delay. (2) To implement the methodology in a prototype decision support tool directed at the brigade level. The key to attaining these goals is selecting a method of modeling ground mobility. Mobility modeling is the focus of the literature review in the next chapter.

# II. Background

## Introduction to Mobility Modeling

Hartman considers three approaches to mobility modeling in high resolution combat simulations: explicit, implicit and network (12:3-24). Each method has a unique way of representing the terrain, determining the movement paths, and analyzing the effect of obstacle emplacements on the movement paths. This chapter will review each method and its advantages and limitations.

## Explicit Mobility Models

Explicit models are of two types: explicit grid and explicit patch (12:3-24).

Explicit Grid Models. Explicit grid models store, for each data point on the battlefield, all the terrain attributes which can effect manuever and speed at that point. Hartman states there are 19 terrain attributes, including: vegetation, soil type, roads, rivers, railroads, bridges and obstacles (12:3-26). The modeler determines the level of resolution by determining how large the grid elements are. Once the terrain attributes are defined, it is necessary to calculate the speed a vehicle can sustain on the terrain. There are two basic ways of determining this speed: table lookup and engineer level calculations.

When table lookup is used, the model accesses the terrain attributes of the point and 'looks up' the limiting speed in a pre-established speed array. The speed array is

the modeler's best estimate of what the limiting speed would be for all possible combinations of terrain factors. In essence the speed array pre-establishes the results of the simulation so it is very important to check how sensitive the model is to changing the limiting speeds.

Engineering level calculations involve detailed modeling of a vehicles movement over different types of terrain. A good example of this type of simulation is the AMC-71 model developed by the Army Material Command (12:3-26). AMC-71 computes numerous mobility characteristics including limiting speed. The model has an extensive database including 19 terrain attributes for each data point and 76 vehicle attributes. The output of AMC-71 can be used to establish the limiting speed arrays.

Explicit Patch Models. Explicit patch models are identical to explicit grid models except in how the data is stored. Recognizing that certain areas on the ground have the same terrain attributes, the data points are batched together into irregularly shaped polygons of uniform type. A data base algorithm is then used to access information on the location, and type of terrain at each point. For areas with large patches of similar terrain explicit patch models can significantly reduce the size of the data base.

Explicit Model Applications. Regardless of whether explicit grid or explicit patch modeling is used, there are currently three ways used to represent the terrain: hexagon,

digital, and functional. The Obstacle Planning System (OPS) uses a hexagonal terrain representation (8). The Dynamic Tactical Simulation (DYNTACS) uses a scalar terrain representation (12:3-8) and the Simulation of Tactical Responses (STAR) model uses a functional terrain representation (14).

Obstacle Planning Simulation. OPS is a single player, two phase, interactive video game designed to teach combat engineer tactics. In Phase 1 the player emplaces an obstacle system, of their own design, representing 8 hours of engineer effort. In Phase 2 the player observes the effect of the obstacle system on an attacking enemy mechanized infantry division against a defending friendly mechanized infantry brigade. Twenty different offensive and defensive scenarios can be played.

OPS has 61 hexagonal map cells. Each cell includes one elevation attribute (ground, slope, or hilltop) and one or more terrain attributes: road, gully, woods, town, swamp, ford, water, clear, and bridge. Each combination of elevation, terrain, and obstacle type results in a 'movement cost' between two terrain cells. The movement cost is established through table lookup and is the fixed amount of time required to move between two cells. OPS runs a discrete event simulation in which combat outcomes are probabilistic and are displayed at the end of the game. OPS has been used by the U.S. Army Engineer School at Ft Belvoir, Virginia.

Dynamic Tactical Simulation. DYNTACS models combat processes from individual to battalion level engagements. The model uses two-sided Monte Carlo simulation and a concept referred to as digitized terrain which provides the elevation, forestation and location of each corner of a 100 meter square (14). DYNTACS divides each square into two equally sized triangles and fits a continuous surface through the three corner points. The resulting terrain is a continuous collection of inclined triangular facets (12:3-7). Each triangle has homogenous assets and the elevation change is linear between the corners. Vegetation and obstacle locations are modeled as overlays to the terrain and are included in determining the limiting speeds across the triangles.

Simulation of Tactical Responses. STAR is a high resolution brigade level combat simulation. In STAR, the terrain is represented by a series of continuous, smooth 'hill mass functions' derived from Gaussian curves (12:3-9). This method substantially reduces the data base storage storing only the parameters of the functions representative of each hill. Terrain characteristics affecting movement are represented as a series of overlays modeled in conjunction with the appropriate terrain location.

## Implicit Mobility Models

Implicit mobility models store the value of a "mobility

multiplier" for each data point rather than the actual terrain attributes. The mobility multiplier considers all the modeled elements which can effect speed such as: terrain, elevation, and obstacle emplacements. The mobility multiplier can be derived using explicit engineer level modeling or established through table lookup.

Dupuy discusses how each item affecting speed can be given a separate mobility multiplier and multiplied together to get an overall mobility multiplier. The overall speed a unit can sustain is calculated by multiplying a nominal base speed capability by a mobility multiplier.

$$\text{Overall Speed} = \text{Base Speed} * \text{Mobility Multiplier} \quad (1)$$

Dupuy shows how the base speed can include both moving unit strength and type, and defensive unit posture and resistance (10:213). The main limitation of implicit modeling is the modeler's inability to establish accurate base speeds and mobility multipliers. As in explicit modeling, implicit modeling results should be checked for sensitivity to changes in the mobility multiplier.

Network Mobility Models

Network models are the latest trend in mobility modeling. In a network model, the nodes represent actual physical locations on the map such as cities, hilltops or road junctions. The arcs are movement paths between the nodes. Each arc and node has mobility characteristics to determine

the size and type of unit which can move along the arc and the speed the unit can sustain along the arc. Again, the limiting speeds can either be explicitly or implicitly determined. Network representation has been used extensively by the Naval Postgraduate School in the AirLand Research Model (ALARM). ALARM models large scale warfare in a European scenario of the type anticipated by AirLand Battle doctrine (22).

Network Modeling Applications. The general applicability of networks to transportation and command and control modeling was demonstrated by Krupenvich (17). Networks were proposed for analyzing movement of units and logistics, identification of key terrain and rear area interdiction processes. The network solution algorithms included: path determinations, flow optimization, and unit location results.

Path determination algorithms can solve the network for the shortest path, the kth shortest path and all possible paths. Flow optimization algorithms determine the maximum flow in the network as well as the minimum cost required to satisfy demand at some terminal node. Location algorithms are used to determine optimal locations for source or sink nodes in an existing network.

Two location algorithms were described by Krupenevich: minimax and minisum. Minimax algorithms find a location which minimizes the maximum distance from the selected

location to the possible demand sites. The minisum algorithm minimizes the average cost of travel, as expressed by the modeler, to any other point on the network.

Krupenvich also proposed using discrete network simulations to represent high resolution modeling of single or multiple units as they move through the network. A discrete event simulation is able to step units through the network locating them in space and time and providing more detail on time delays and movement conflicts between units.

Network usage was expanded by Fletcher to generate avenues of approach (11). In this research, flow rates based on terrain type were used to develop zones of action in the network. These zones of action were used to determine arcs which could sustain advancing battalions. From these zones of action a connected network was formed to determine the movement paths of a regimental size unit. Fletcher's analysis is similar to corridor movement modeling discussed by Hartman (13:3-9).

Fletcher also discussed the planning of the phased commitment of ground combat forces. Command and control in the model was based on a projection of the moving unit's power using the Generalized Value System (GVS) proposed by Parry and Schoenstadt (22) and further discussed by Kilmer (16). The advancing unit's power was then used to determine defensive mission feasibility, decision points, and feasible courses of action. Fletcher's research resulted in a

methodology for generating a defensive plan for a brigade size unit facing a motorized rifle division in the attack.

In 1987, Choi demonstrated how a high resolution data base could be used in conjunction with a user established network of arcs and nodes (3). The data base used 100 meter grid square data to determine the arc widths, possible speeds, and flow rates for both mounted and dismounted units moving in either single or multiple columns. Both the shortest path and the minimum time path were found using standard network algorithms.

Several other network analysis concepts have been demonstrated in work done in scheduling airstrikes. Mustin (20) and Nugent (21) proposed algorithms for allocating airstrikes on the minimum feasible cut set in a transportation network. Wollmer (26) established a cost based on network flow and considered planning interdiction to maximize the costs associated with maintaining a minimum level of flow in the network.

In addition to standard network algorithms, artificial intelligence search techniques can be used to solve the network. Brown discussed the use of depth first search to determine minimum time paths within an Organization of the Joint Chiefs of Staff contingency planning model (2). The research showed that a depth-first search produced shortest path solutions very quickly even for networks at maximum

allowable size and Brown suggested further research into using breadth-first and best-first search techniques (23).

Network Countermobility Applications. Kazimer proposes an algorithm for obstacle emplacement using network modeling (15). First the network is analyzed to determine the minimum time path. Then an engineer obstacle, which has the largest ratio of delay time to 'util' cost, is selected to interdict this minimum time path. The 'util' cost is a common unit of measurement combining both the time to arrive at the target and the work time, equipment and manpower needed at the target.

The obstacle selected is compared with a pre-established table of obstacles which are appropriate for differing types of terrain. Once the obstacle is placed, the network is reevaluated to determine the new minimum time path and this cycle is repeated until all available resources are used. The resources needed for each obstacle are standard packages similar to those used by Slattery (25:26). Kazimer used Branch and Bound to search the space of feasible solutions and to find the optimal obstacle emplacements on a small network model.

Craig modified and implemented Kazimer's obstacle emplacement concepts on an actual section of ground in the Federal Republic of Germany for a brigade in a defensive position (4). Craig used breadth first search to solve the single source network problem and used a Lanchester linear

law formulation to establish the outcomes of unit engagements on the arcs. The results of the final optimal obstacle emplacement agreed with the results obtained by a U.S. Army Command and General Staff College school's analysis of this area. In both Craig and Kazimer's works, arcs in the network are fixed so obstacle emplacements are modeled as degradations to speed along the arc. Units can bypass obstacles but they must do so within the width of the arc.

McLaughlin integrated the placement of manuever units and countermobility assets (18). Prior work had separate algorithms for movement and engineer asset placement: this caused some obstacles to be placed without being covered by fire. McLaughlin proposed algorithms which consider the potential of terrain for countermobility operations to enter into the arc evaluation process insuring obstacles and friendly units are placed on the same arc.

McLaughlin also explored three criteria for avenue of approach generation: minimum time, best road, and best flow. The minimum time path was usually the shortest path and frequently contained choke points for defensive planning. The best road algorithm found the avenue with the best and widest road surfaces. These avenues were excellent for determining routes for movement of logistic assets. The best flow path developed a route with the best off-road maneuverability.

When evaluating an arc from the countermobility standpoint, McLaughlin considered both the effect of previous obstacle emplacements and the potential of future obstacle placements. The effect of countermobility operations were measured in the amount of delay time the moving unit incurred. The time delay was found to be dependent on the enemy unit's composition, the availability of engineer resources, and the defender's coverage of the obstacle by fire.

Network Modeling Discussion. Although network analysis has been extensively applied to analyzing movement, it is not without fault. To date, the modeler must determine the location of the arcs and nodes in the network. Movement is restricted to these arcs. This limitation may not be a problem when modeling vehicle movement on road networks. However, cross country movement requires a very high resolution network to insure off road movement is considered and, at the size required, network solution algorithms run very slowly. In some cases to reduce the number of arcs in a network, arcs which do not meet a minimum flow rate are eliminated from consideration. This assumption may not be realistic for analyzing cross country movement since even very poor arcs might sustain enough flow to influence the outcome of the simulation.

Speed within an arc is usually the average of the speeds which can be attained across the data points on the arc,

this aggregation may not accurately reflect the speed. Additionally, network algorithms require a source and a sink node. In other words, there must be a start and a finish point. In the movement of supplies or transportation assets, there may actually be a depot and a receiving unit located at nodes in the network. However, in cross country movement there may be several start points and several objectives making movement difficult to analyze.

Lastly, the results of network analysis can also be suspect. Maximum flow determination is based on an equilibrium situation in a closed network. Therefore it is difficult to interdict the arcs. Normally, the simulation is rerun from the source node with the interdiction in place. It is clear that, network analysis is usually only as good as the modeler's ability to identify: the location of the arcs and nodes, the location of the source and sink nodes, and the level of resolution.

# III. Methodology

## Methodology Overview

The manuever area selected for study is an approximately 10 kilometers by 10 kilometers region located east of the city of Fulda in the Federal Republic of Germany. This size area is appropriate for a U.S. brigade in the defense against two motorized rifle regiments in the offense.

The end result of this analysis is a working prototype which allows the user to model the study area, determine the avenues of approach, and analyze the effect of engineer obstacles on the avenues of approach. The prototype provides the following:

(1) A user interface to provide parameters used in the program and specify files to be used and generated.

(2) An elevation contour map for the area derived from Digital Terrain Elevation Data.

(3) A terrain map for the area.

(4) A representation of the speeds the enemy can be expected to sustain in the movement area.

(5) A representation of the movement paths and times for the area.

(6) An interactive method for placing obstacles on the movement paths to see the effect on the movement paths and times to cross through the area.

## Digital Terrain Elevation Data

Digital Terrain Elevation Data (DTED) is produced by the Defense Mapping Agency Aerospace Center (DMAAC) (5). DTED provides a digital map of elevations along the face of the earth. In a standard DTED file, the interval between elevation values is three arc-seconds or approximately 300 feet (93 meters). The location of terrain in DTED is referenced to the World Geodetic System, and the terrain elevation values are relative to Mean Sea Level.

Each DTED file contains the elevations for a one-degree square of the earth's surface and records within the file are relative to the latitude and longitude of the southwest corner of that square. Elevation values within the file are read from south to north then west to east.

DTED data for this model was obtained with the assistance of Aeronautical Services Division personnel in the Avionics Lab at Wright-Patterson Air Force Base (24). The elevations within the study area range between 331 meters and 771 meters mean sea level. Figures 7 - 14, in the appendix, display contour maps for the elevations found in the study area. In the prototype the graphics screens are in color but for purposes of presentation in this thesis black and white composite drawings are included.

## Cultural Terrain Data

Terrain within the area was divided into eight categories as shown in Table I. Each category is identified with an

integer code. Figures 15 - 19, in the appendix, show the relative proportions and locations of the terrain types in the study area. These screens help the user see the location of population centers the road networks. Also, the user can get a feel for how water and swamp areas may effect enemy movement.

Table I

Cultural Terrain Types

| Integer Code | Terrain Type |
|---|---|
| 1 | Forest |
| 2 | Urban |
| 3 | Marsh or Swamp |
| 4 | Water |
| 5 | Open or Shrubs |
| 6 | Autobahn - 4 Lane Road |
| 7 | Improved Road - 3 or 2 Lane Road |
| 8 | Unimproved Dirt Road or Lane |

## User Interface

Since this prototype is designed with the user in mind a startup menu was created as shown in Figure 1. This menu contains many of the variables discussed in the following sections and allows the user to change the default parameters in the program before conducting the analysis.

## Movement Directions

Within the movement area there are 10,000 data points. Each point represents a 100 meter by 100 meter square of uniform elevation and terrain type. Movement through the

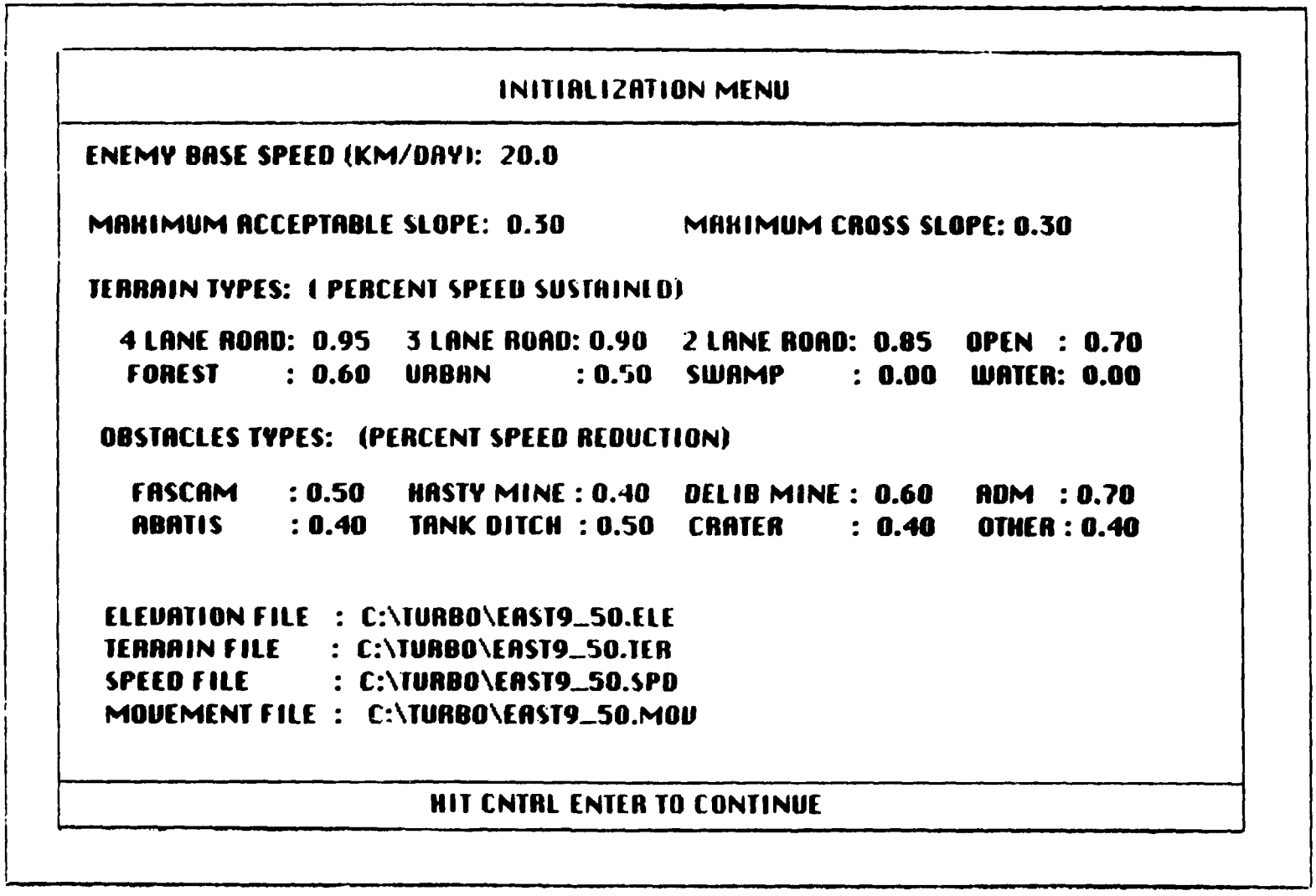


Figure 1. Initialization Menu (Default Values Shown)

data points is from east to west. From any location, there are three possible movement directions: straight ahead, right diagonal and left diagonal, as shown in Figure 2.

Speed Determinations

The speed between any two data points is dependent on a base speed the enemy is expected to sustain, degraded by the terrain and elevations of the points.

Speed = Base Speed * Speed Degrade Due to Cross Slope
* Speed Degrade Due to Elevation
* Speed Degrade Due to Terrain (2)

Base Speed. The base speed is the estimated overall speed that can be sustained in the movement area. The base speed can be estimated considering the type of moving unit and the type of resistance and posture of the defending unit. The base speed can be in any unit of measurement, this analysis uses kilometers per day and the base speed is assumed to be in a flat, open, hard surface environment.

Speed Degrade Due to Cross Slope. Cross slope is defined as the slope perpendicular to the direction of movement. Checking the cross slope insures that the points are not on the side of a hill which would cause vehicles to roll over if they attempted to move between the points. Figure 3 shows moving between F and E requires checking the slope between: F-I, F-C, E-H, and E-B. But moving from E to G, cross slopes E-I, E-A, G-D and D-J are checked.

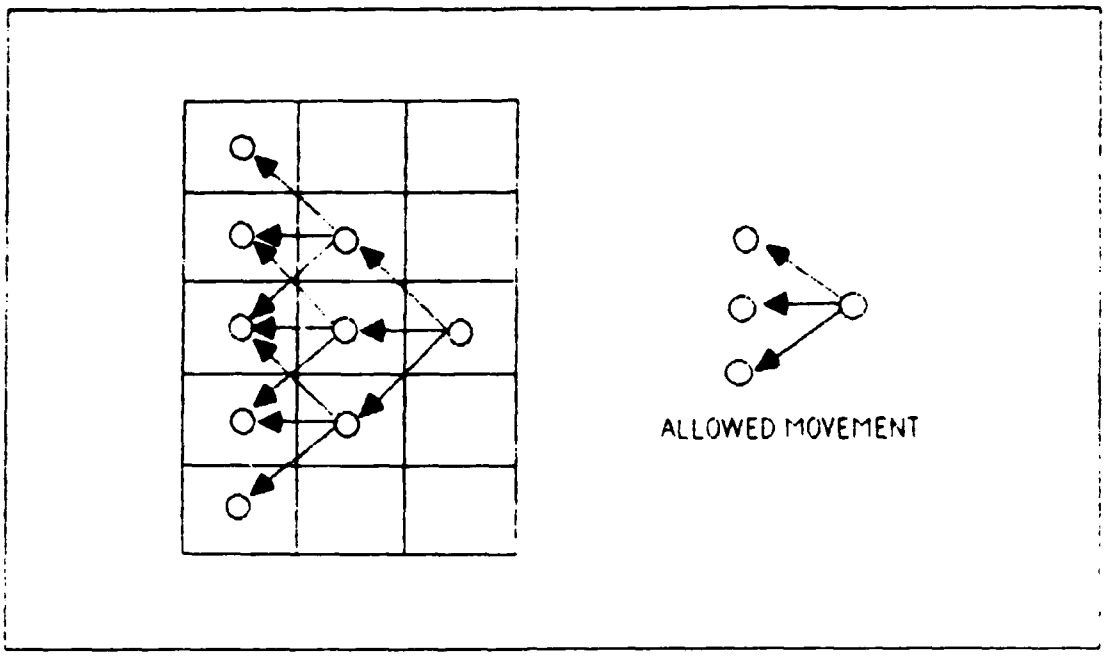


Figure 2. Allowable Movement Directions

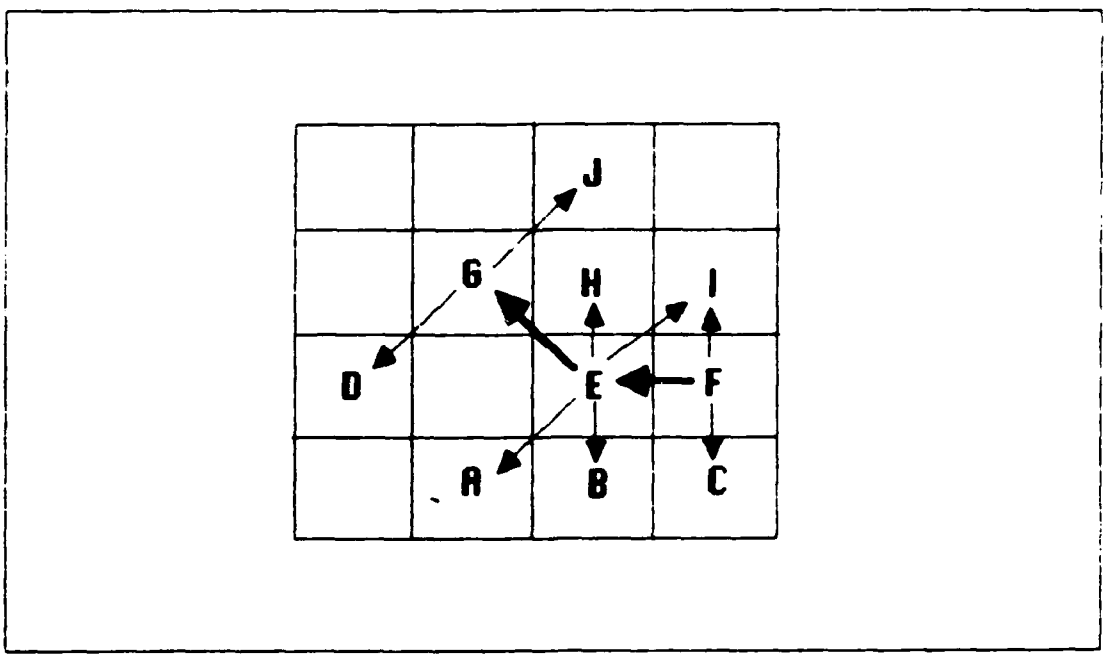


Figure 3. Cross Slope Check

The user sets the maximum cross slope that can be traversed, the default is 30%. The speed degradation due to cross slope is either 1 or 0, indicating that movement is either acceptable or not. Since a vehicle can manuever to the left or right of a point, both left and right cross slopes must be unacceptable for the cross slope degrade to be zero.

Speed Degrade Due to Elevation. The slope in the direction of movement is modeled as a linear degradation of the base speed. For example a 20% slope results in a 20% base speed degradation. The user can adjust the maximum acceptable slope for wheeled or tracked vehicles. The default is 30% or less.

Speed Degrade Due to Terrain. The terrain of the two points also affects the base speed. Table II shows how the speed is reduced based on terrain type.

Table II

Terrain Effects on Speed

| Terrain Type | Base Speed Reduction (In Percent) |
|---|---|
| Forest | 40 |
| Urban | 50 |
| Swamp | 100 |
| Water | 100 |
| Open | 30 |
| Autobahn | 5 |
| Improved Road | 10 |
| Unimproved Road | 15 |

In this analysis swamps and water points are assumed to be impassable. However, the user can adjust these defaults

for different scenarios. Impassable terrain at either point degrades the speed to zero. Passable terrain is averaged to get the overall terrain degradation. In other words, leaving a forest point and entering an open point would result in a 35% base speed reduction between the two points.

Speed Levels. There are six levels of speed as shown in Table IV. Six levels were arbitrarily chosen for the best graphical display.

Table III

Speed Levels

| Speed Level | Speed Sustained (In Percent) |
|---|---|
| Fast | 84 - 100 |
| Acceptable | 67 - 83 |
| OK | 50 - 66 |
| Slow | 33 - 49 |
| Marginal | 17 - 32 |
| No Go | 0 - 16 |

Figures 20 - 23, in the appendix, show how the points in the movement area are dispersed according to speed level. Comparing the acceptable and fast points to the existing road network shows a distinct correlation between the two. Likewise, there is a correlation between impossible and marginally possible points and areas of swamp or water.

Movement Paths

Although the speed calculations give basic information on the proportion of high speed points and their location, the calculations look only one step ahead. In reality,

movement is conducted after looking across the entire movement area to the objective and selecting the best route. Some slow or marginal points may be along the selected path if they connect to faster points.

To consider this forward looking movement planning, an artificial intelligence technique called A* (pronounced A-Star) search is used (23:80). A* search is an informed best first search designed to find an optimal path between a start state and a goal state, given the possible connections between the states.

For the movement area, any data point on the eastern edge of the area is a possible start point (100 possible start points). Any point on the western edge of the area is a possible goal state (100 possible goal states). The movement directions provide connected paths from the start states to the goal states. Since all start points are feasible, there are 100 optimal paths to be solved. An optimal path is defined as the minimum time path between the start point and any goal state.

In order to understand A* search, certain basic definitions are helpful.

State. Each node (data point) in the search space is a state defined by an x and y coordinate. The top left-hand node is 1,1 and the bottom right-hand node is 100,100. A state is used to determine where the search is or has been in the problem space.

Parent. The parent of any node is simply the predecessor node of the current state. Since each node has three possible parents, it is the parent along the current path that is important.

Child. Since there are three possible movement directions each node has three possible children.

G. G is the known cost of getting to the current state. Cost in the problem space is time. Time is calculated using the known speed and the distance between the two points. The overall time to get to the current state is found by summing G along the path.

F. F is an estimate of what the cost will be to get from the current state to the goal state. A* search produces an optimal solution as long as F underestimates the actual cost to the goal. By getting F as close as possible to the actual cost (without exceeding) the search proceeds more quickly. For the movement area, F is the actual time already accrued (G) plus half of the number of steps remaining to get to the goal, times 0.50. The multiplier can be varied to minimize the amount of time the search takes; 0.50 was found to be the best. Since every path is 100 steps long, F is easily determined.

$$F = G + ((100 - Current\_X\_State) * 0.50)) \quad (3)$$

Open List. The open list is one of two lists maintained during the search. The open list contains all the

nodes which have been visited but whose children have not yet been visited.

Closed List. The closed list is a list of nodes which have been visited and whose children have been visited. These nodes have a known cost (G) and the minimum time path will be selected from among these nodes.

Best Node. The best node is the node on the open list with the minimum value of F. This is the node that has been selected to be placed on the closed list and its children examined. The best node is the last node in the current minimum time path.

The A* search algorithm proceeds as follows.

Step 1. Put the start node on the open list and set the closed list to nil.

Step 2. Let the best node be the node on the open list for which F is a minimum.

Step 3. Move the best node from the open list to the closed list.

Step 4. Generate each child of the best node and do following:

If the node is not on the open list or the closed list put the node on the open list.

Else if the node is on either the open list or closed list and the current F is better than the old version, update the list. If the current F is not better than the old version, discard the child.

Step 5. Continue expanding the tree until the best node is at the goal state. Trace back through the parents to determine the optimal path. Each path in this analysis is calculated independently from the start point without regard for previous optimal paths.

For the start point at the middle of the eastern edge of the study area, the search tree contains 7301 nodes. A* search finds the optimal path for this node in approximately 4.6 seconds. All 100 paths take approximately 4 minutes and 29 seconds to calculate. Figure 4 shows the minimum time path (in black) for each start point to the goal state. Movement is from the right to the left. The hollow line is the best time path through the area. It requires 17.64 hours.

Directly to the right of the movement paths, the bar graph shows, for each start point, the overall path time relative to the other start points. The slowest time path to cross through the area is 20.09 hours. The mean crossing time through the area is 19.82 hours with a standard deviation of 0.97 hours.

Interdiction

Once the minimum time paths are calculated and the best path is identified, the effect of obstacle placement can be studied. Eight types of interdiction obstacles are permitted. These obstacles do not specifically pertain to any approved doctrinal obstacles but reflect, in general, the

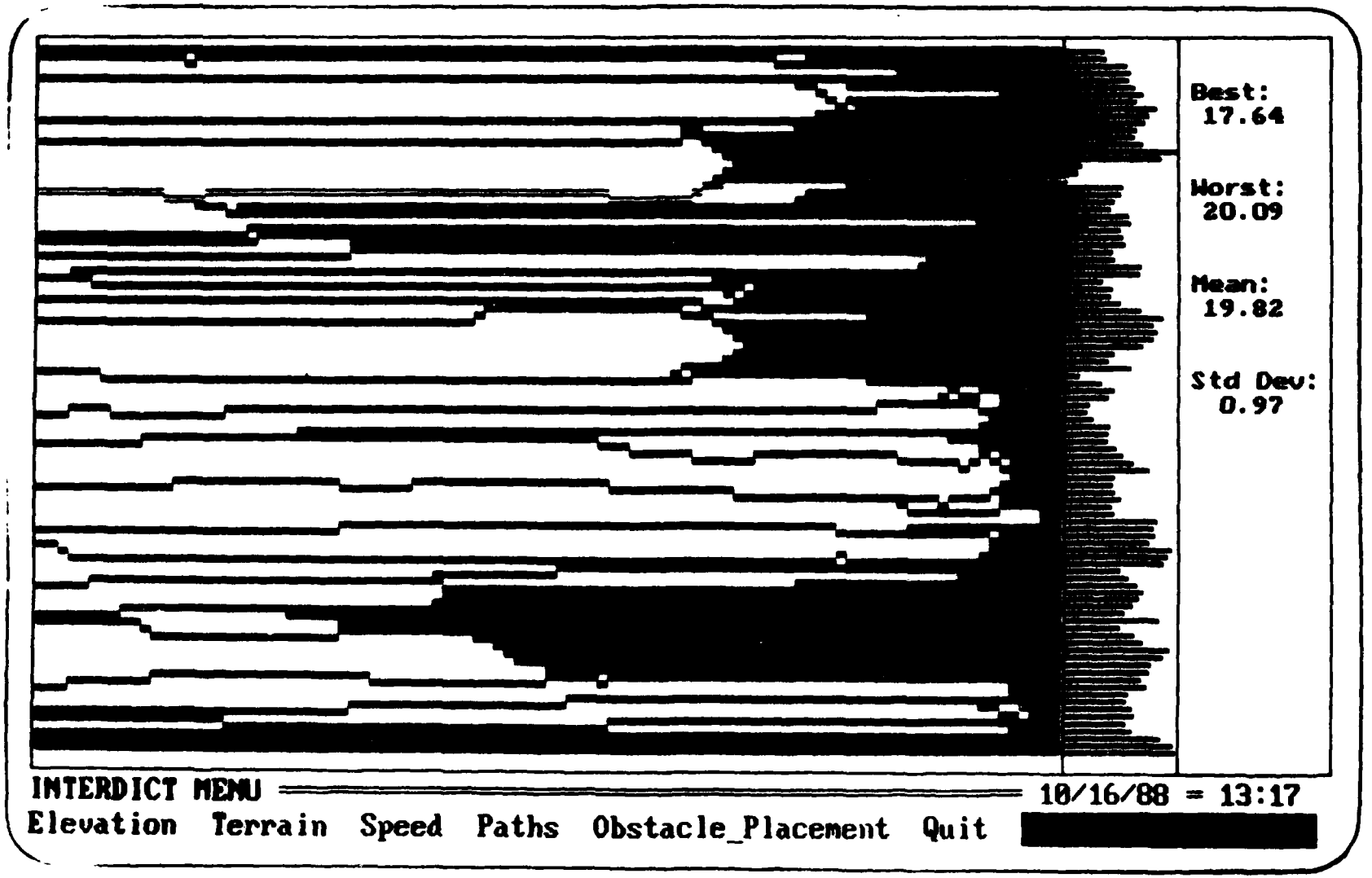


Figure 4. Movement Routes With Fastest Path (Initial)

interdictions the user wishes to place. All obstacles are assumed to effectively interdict each 100 meter by 100 meter data point in which they are placed.

Scatterable Minefields. Scatterable minefields use mines which are "remotely delivered by ground systems, artillery, helicopter or high performance aircraft" (6:29). Scatterable minefields are designed to self-destruct after a set period of time.

Conventional Minefields. Conventional minefields are not designed to self-destruct and are normally "directly emplaced by hand or by mechanical mine planting equipment" (6:29). This analysis considers two types of conventional minefields: hasty and deliberate.

Atomic Demolition Munitions (ADM). These munitions are used to "destroy targets considered difficult or impossible to destroy by other means" (6:146). Typical ADM targets are: defiles and tunnels, bridges, stream cratering, dams, dikes, and airfields (6:147).

Anti-Tank Ditch. Anti-Tank Ditches can be of either triangular or rectangular design. Additionally, these ditches can be mined and wired (6:118).

Crater. Craters are normally placed on "roads or other high speed routes" (6:107). There are four types of craters: hasty road crater, deliberate road crater, relieved face road crater and angled road crater (6:107-111). This analysis groups all the types of craters into one category.

Abatis. An abatis is a tree fall in the direction of enemy movement, usually reinforced with mines and wire. An abatis is particularly effective in "heavily-wooded areas with few roads or trails" (6:126).

Other. This option allows the user to select an interdiction method not previously mentioned, possibly a field expedient obstacle such as log hurdles.

Table IV shows the speed reduction due to each type of interdiction. When an interdiction method is selected and placed, the speed at the affected data points is reduced according to Table IV. The values in Table IV are defaults and can be adjusted by the user. Once the speed at the effected points are adjusted any path which goes through these points is recalculated to see if a new path is selected.

Table IV

Speed Reduction Due to Obstacle Emplacement

| Obstacle Type | Speed Reduction (In Percent) |
|---|---|
| Scatterable Minefield | 50 |
| Hasty Conventional Minefield | 40 |
| Deliberate Conventional Minefield | 60 |
| Atomic Demolition Munitions | 90 |
| Anti-Tank Ditch | 40 |
| Abatis | 50 |
| Crater | 30 |
| Other | 40 |

Since the study area is only 10 kilometers wide it is assumed the moving unit will have intelligence regarding the obstacles and will make decisions based on this knowledge. Figure 5 shows the effect of placing a 200 meter by 200 meter deliberate conventional minefield. In this case the path goes around the obstacle and the mean time through the area has increased by three minutes. Figure 6 shows a less difficult obstacle which the enemy chose to breach. Again the best path has changed but the average time through the area has increased by approximately one minute.

Different obstacle emplacements can be evaluated using this methodology. The user can select obstacle emplacements which provide a desired overall delay time through the area. The prototype can be used to determine the level of obstacle needed to force a breach or a bypass and the time delay achieved by either decision. Combinations of obstacles can be placed which will force the enemy to move through or exit certain areas.

By reducing the standard deviation the user can force the majority of the enemy to cross the area together. Or by increasing the standard deviation the user achieves a larger window of crossing times through the area. Lastly, the singular importance of any one obstacle can now be determined so obstacles can be prioritized by how critical they are to the barrier plan.

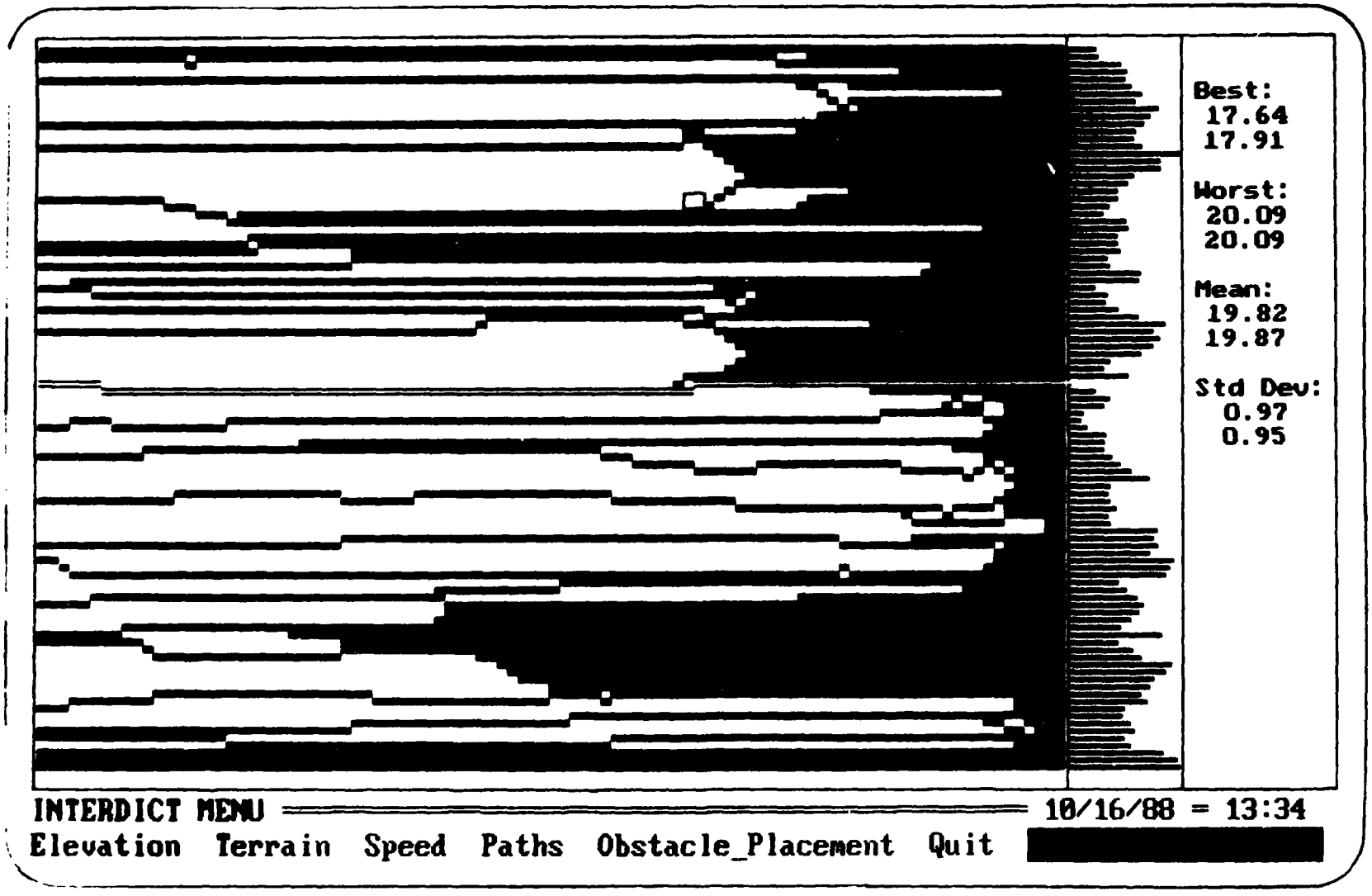


Figure 5. Obstacle Placement Resulting in Bypass

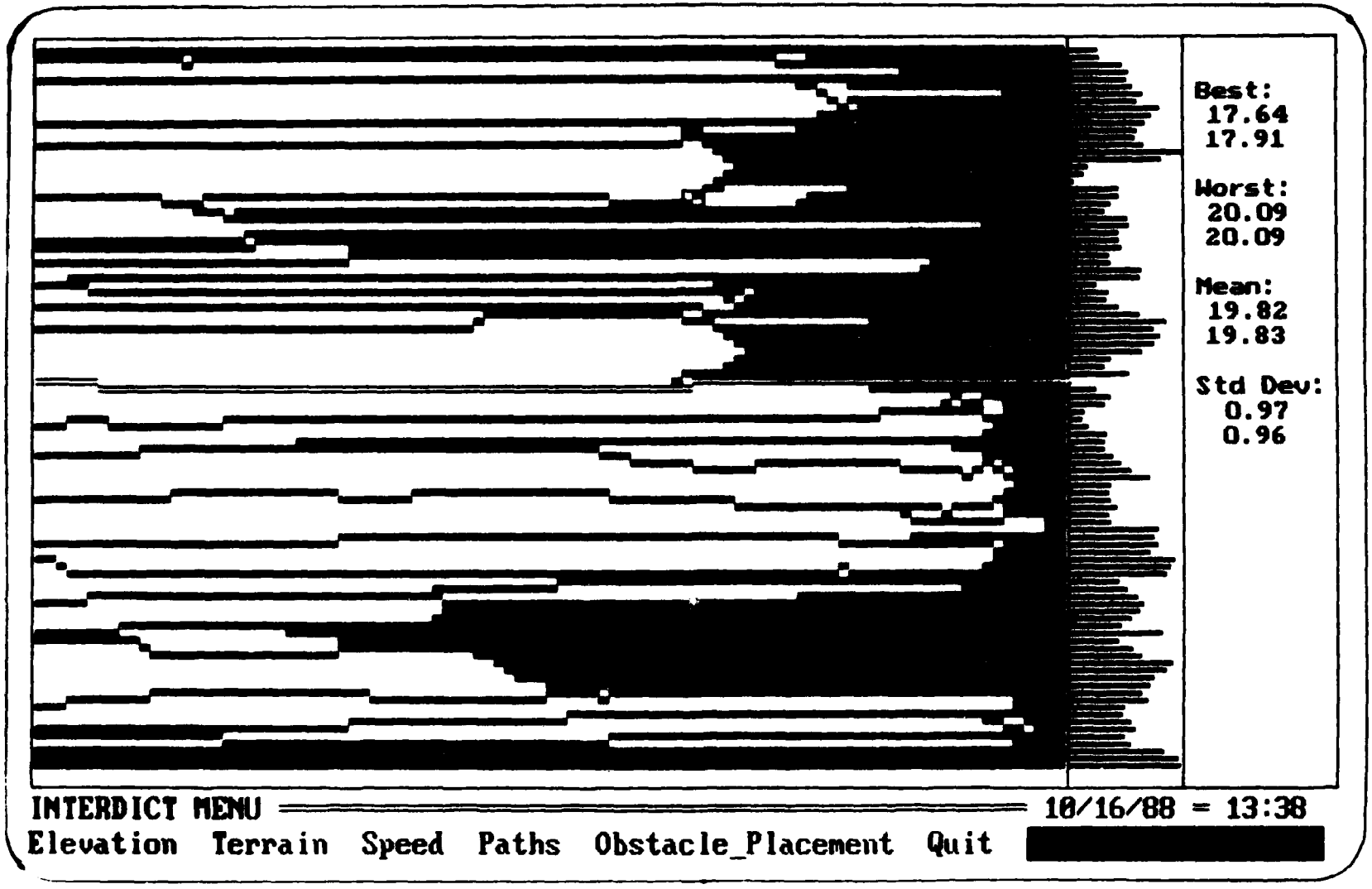


Figure 6. Obstacle Placement Resulting in Breach

# IV. Findings and Recommendations

## Findings

This thesis has shown an effective methodology for quantifying and presenting to the decision maker the effect of engineer obstacles on delay and channelization. The methodology builds upon three basic items of information: elevation, terrain, and allowable movement directions. The speed between any two points is explicitly calculated using: base speed, cross slope check, and speed reduction due to slope and terrain. To insure that movement is planned looking across the entire movement area, A* search is used to find the minimum time path across the study area. The paths can be interdicted and the resulting effect on movement and time determined. This thesis has also shown that the methodology can be implemented in a decision support tool which runs in real time and allows the user to interactively place obstacles and determine their effects.

## Recommendations

This methodology and prototype can be expanded in many directions.

Data Base Resolution. The data base for the model could be expanded to include bridges, dams and other terrain features for which specific obstacles can be used. Additionally, factors such as weather, visibility, soil type, moving unit type, defending unit type and posture could be added to better model the limiting speed of the moving unit. Obstac-

les could be expanded to correspond to specific doctrinal types including their manpower and logistic requirements so resource limits would be automatically tracked. These requirements could be modeled using standard obstacle resource allocation packages.

Command and Control Issues. The movement algorithm could be expanded to take into account how far a unit can move left and or right of its start points. This would keep units from traversing into other units areas and better model the command and control aspects of the movement. Also the lateral distance between units could be checked to model the fact that units may slow down to keep adjacent units in sight. The methodology currently has no restriction on the number of entities that can occupy a point or path. A discrete event simulation could be used to locate each entity in time and incorporate decisions on how the units will move to avoid massing at any one point or along any path. The movement of entities through time could be graphically depicted by showing each unit's position in time as it moves through the area. The model currently has the moving unit uniformly moving into the terrain, this could be changed to a random or user defined distribution. Likewise all exiting points are feasible, this could be changed to force the enemy to exit within a certain group of points by placing impassable terrain on adjacent points. Lastly, the

user could be allowed to select an objective area and the movement paths run specifically to those points.

Expert Obstacle Placement. A wide variety of algorithms could be developed to allow the program to select the best combination of obstacles for the given area under study. These algorithms could optimize the emplacement based on a desired time delay, minimum flow through the points or resource constraints. An Standard Operating Procedure table showing the appropriate type of obstacle for a given area could be used to select the best obstacle. The user could specify a preference for the types of obstacles in case of a tie. Scheduling algorithms could be added to provide a work schedule for the expert or the user's obstacle emplacement. Corp and division directed obstacles could also be added before any obstacles are placed. Once the emplacement is complete the user could review the selection and make adjustments as necessary.

Transportation Network Modeling. This thesis is primarily directed toward modeling cross country movement; however, movement could be restricted to the road network by making off road trafficability infeasible. The model could then be used to look at transportation networks for wheeled vehicles and logistic movement.

Friendly Unit Movement. The methodology used studies enemy movement and friendly countermobility missions. This could easily be turned around to model friendly unit

movement based on enemy obstacle emplacements. The model would show whether it is better for friendly forces to breach or bypass an enemy obstacle and what effect engineer mobility tasks might have. The model could be expanded to allow movement into or out of the area from any direction.

Generalized Value System. The results of this research could be directly applied in the Generalized Value System. Time delays and channelization results could be used to determine the loss of firepower a unit sustains when moving across an obstacle.

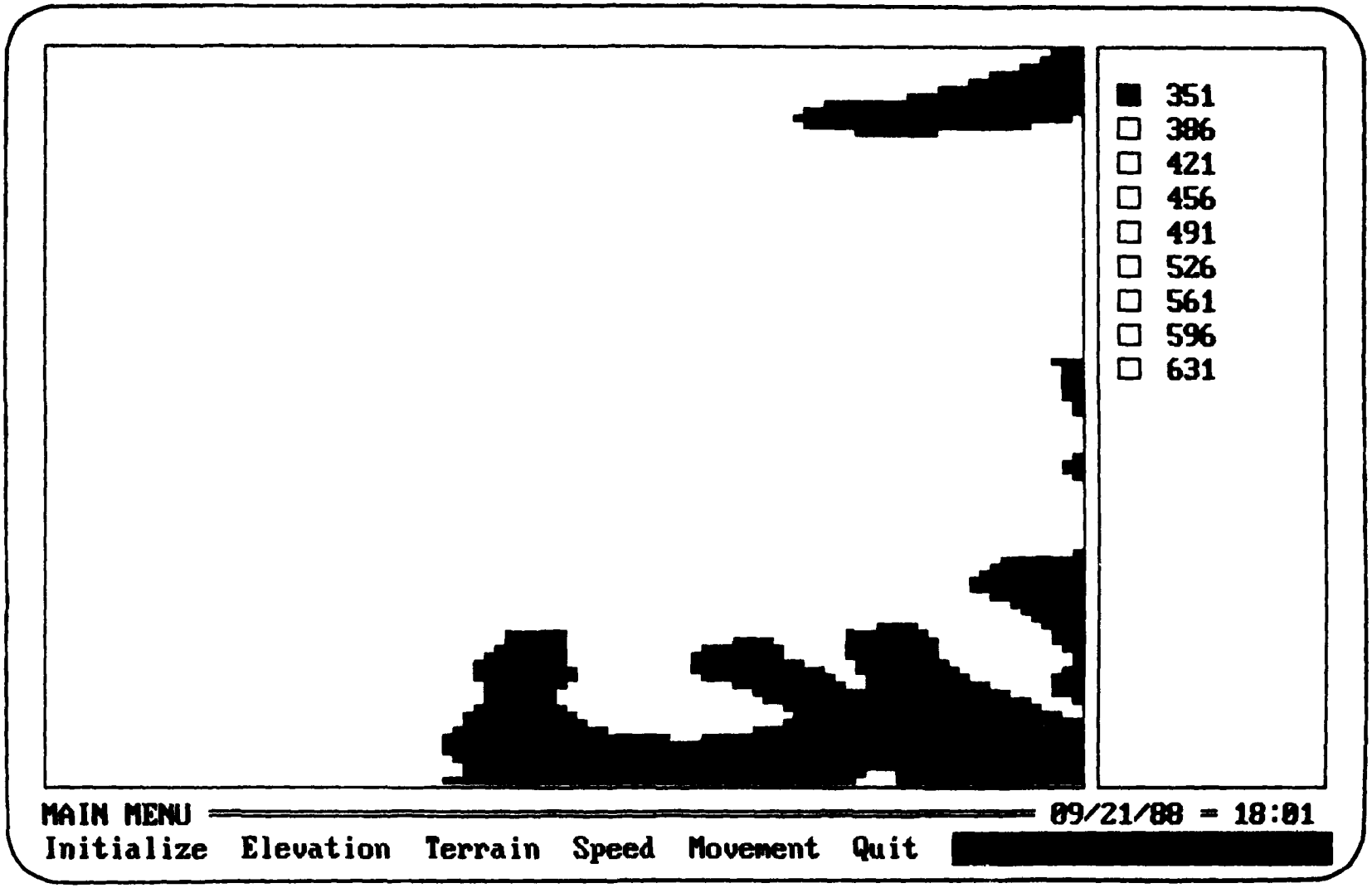


Figure 7. Elevation Map - 331 to 385 Meters

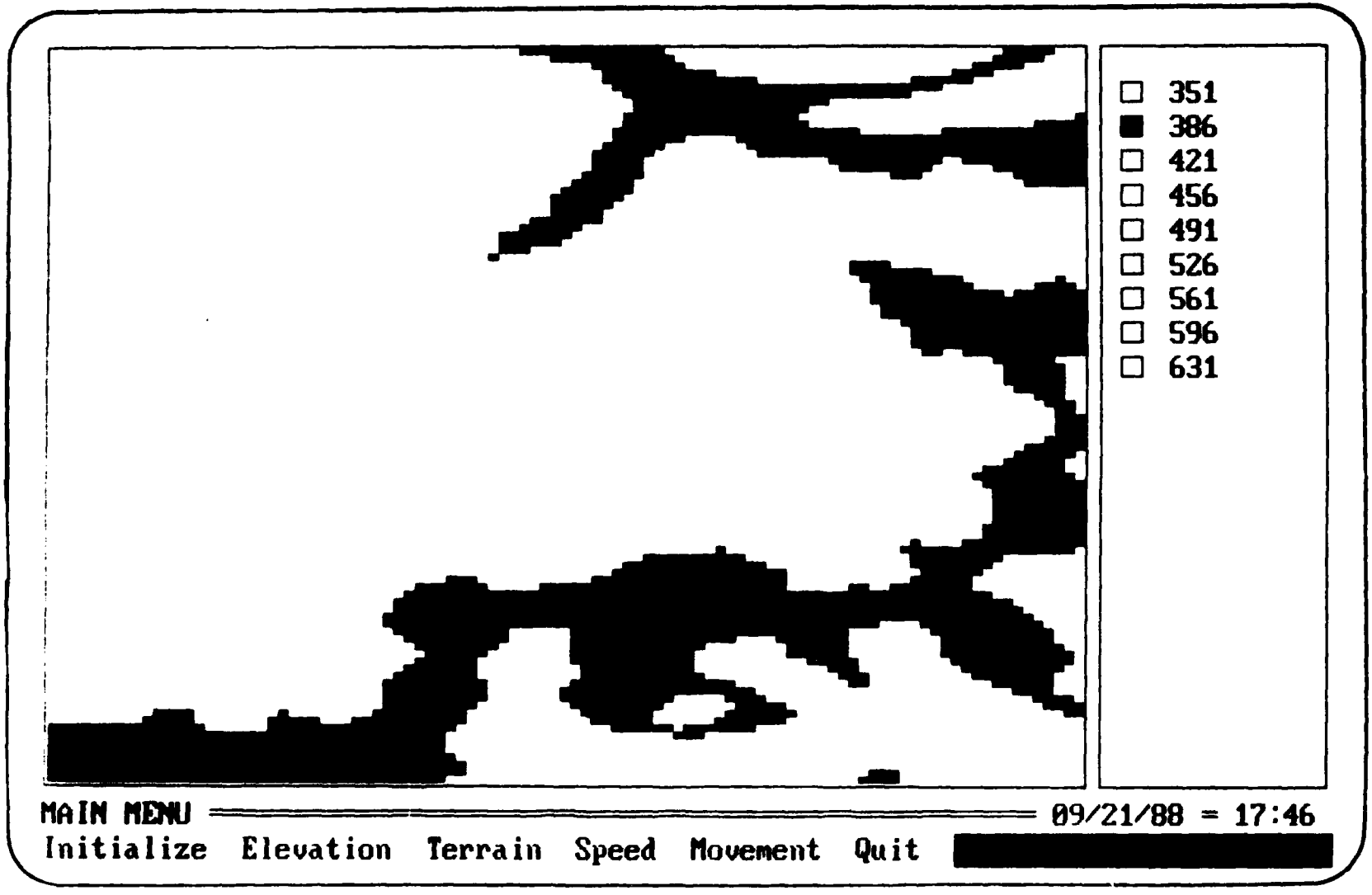


Figure 8. Elevation Map - 386 to 420 Meters

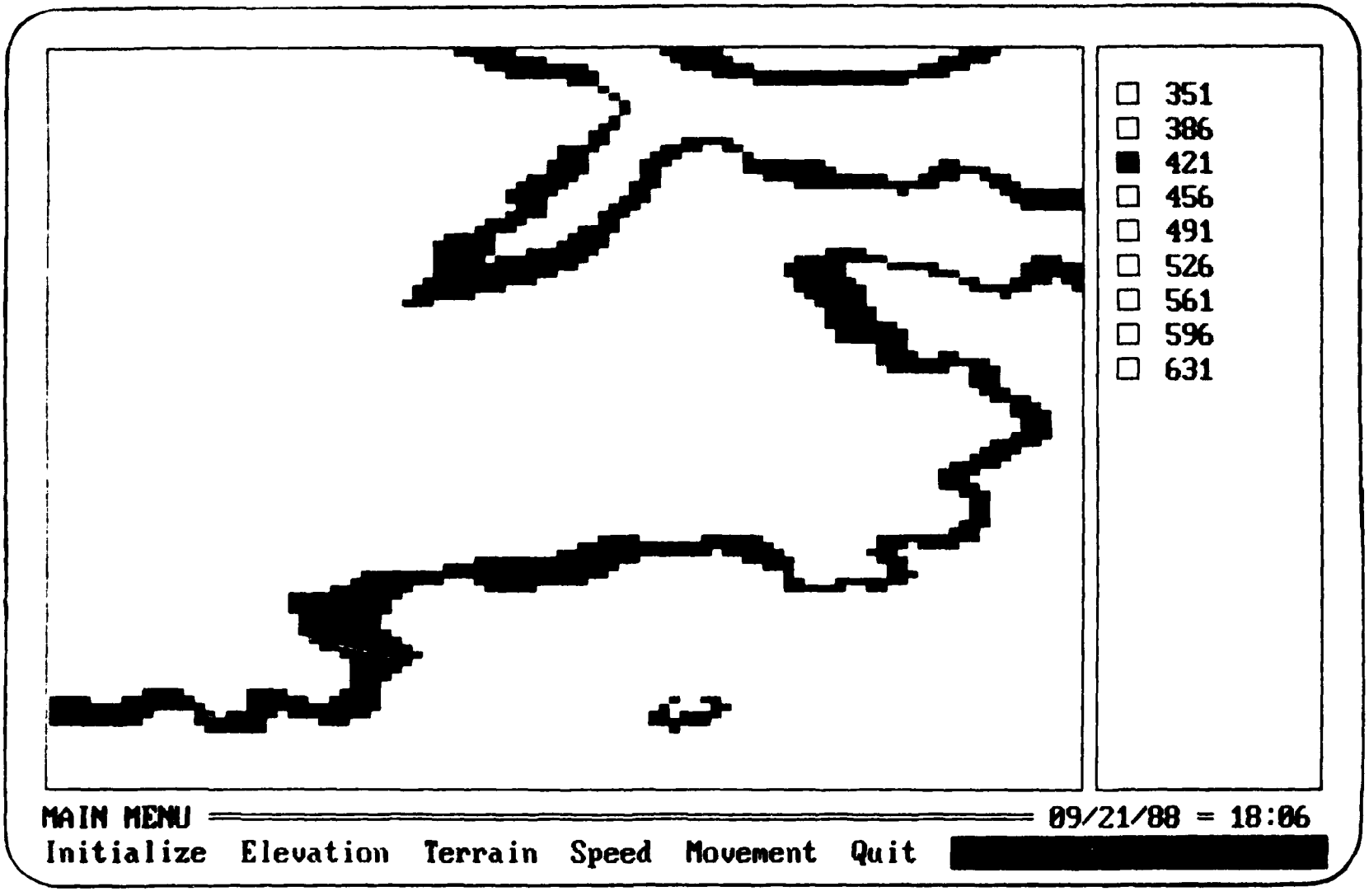


Figure 9. Elevation Map - 421 to 455 Meters

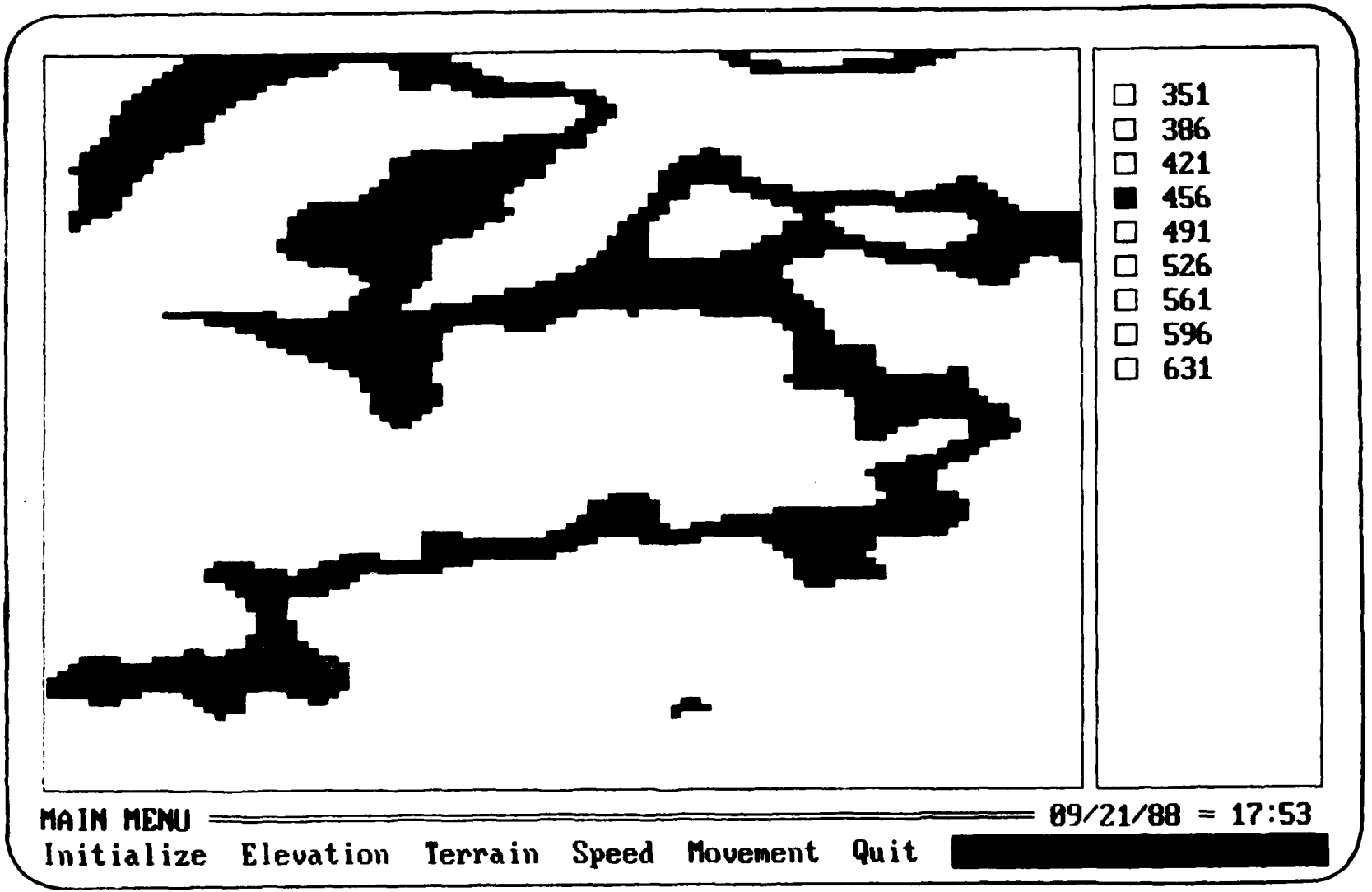


Figure 10. Elevation Map - 456 to 490 Meters

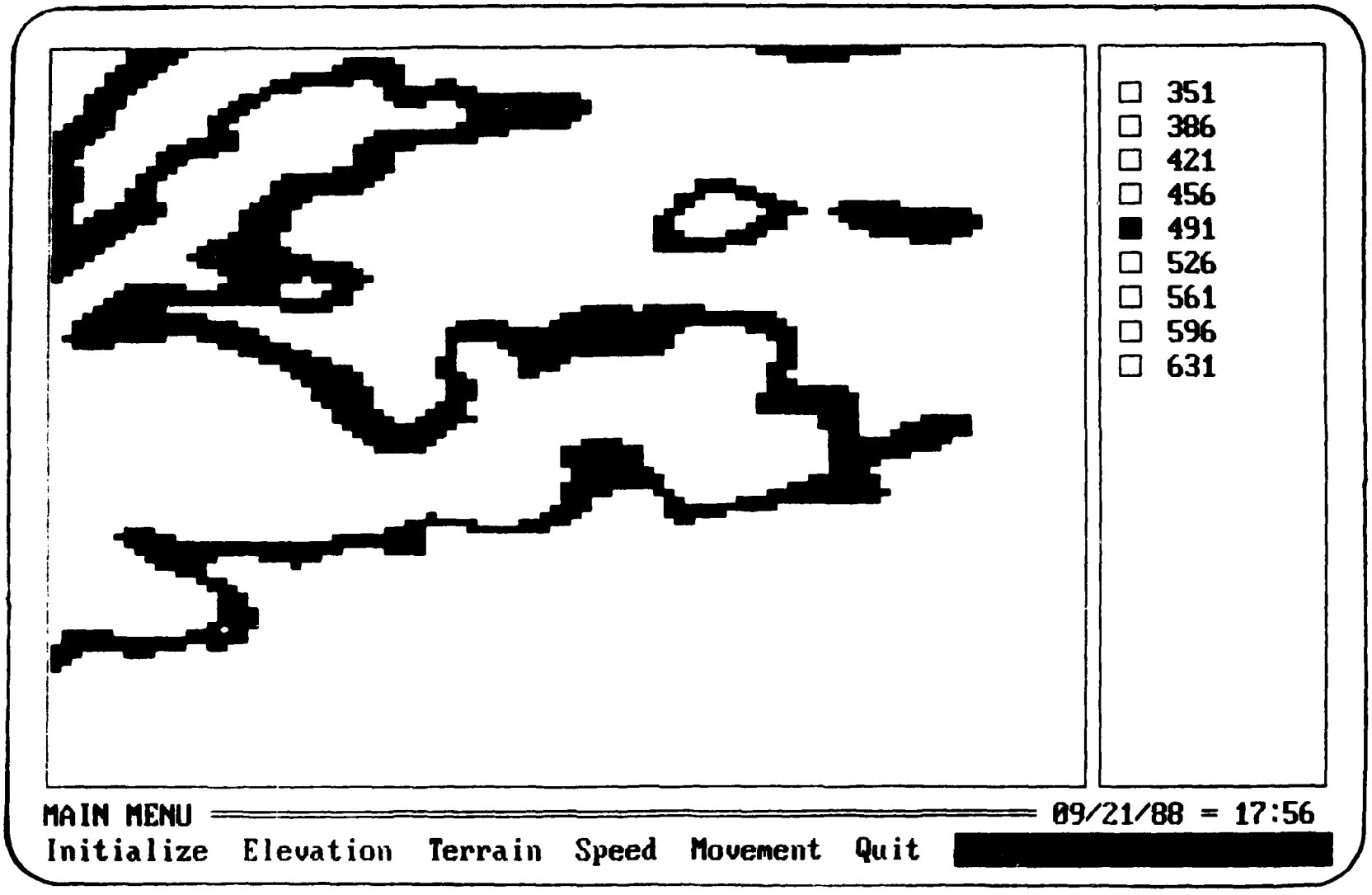


Figure 11. Elevation Map - 491 to 525 Meters

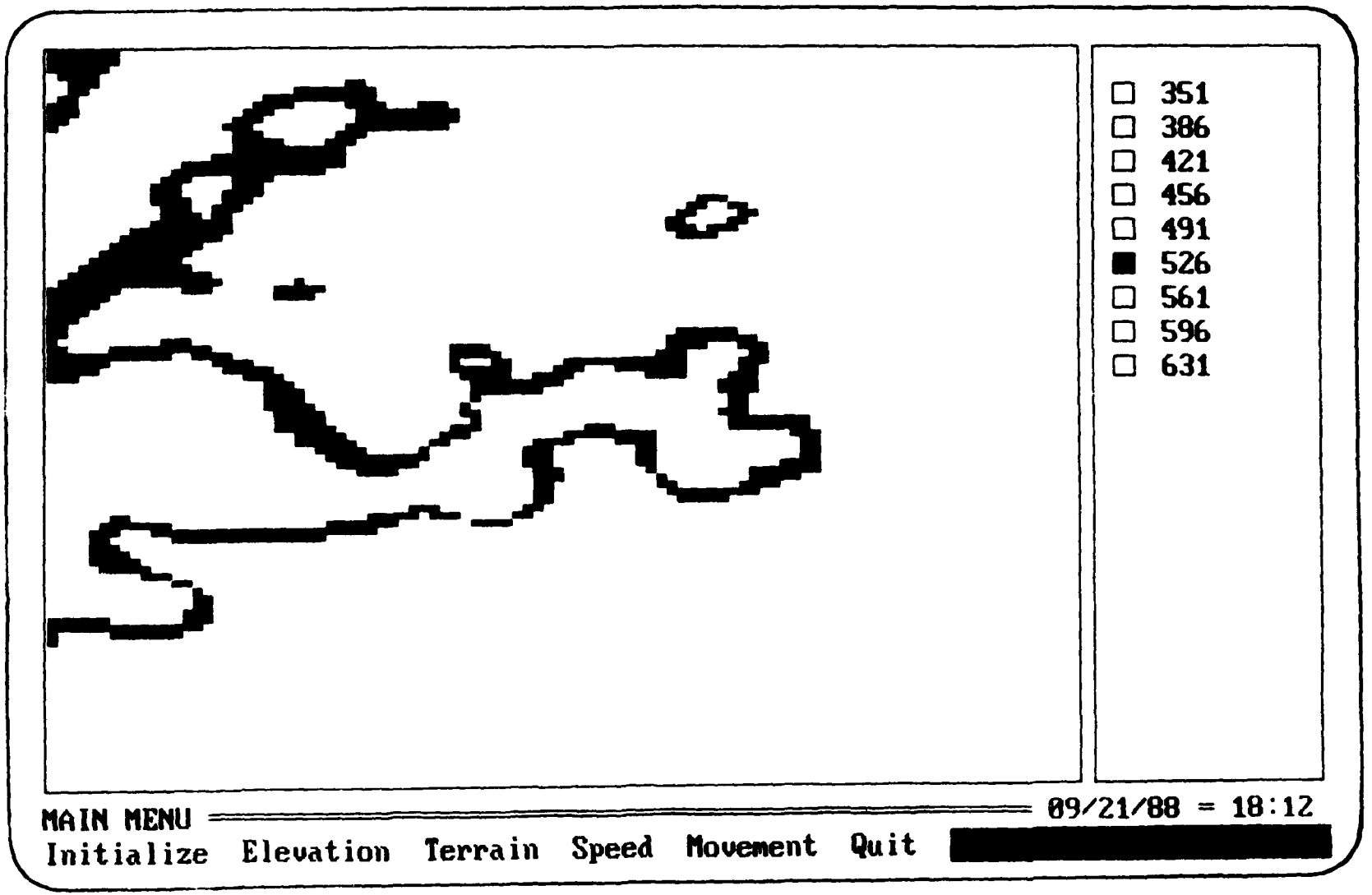


Figure 12. Elevation Map - 526 to 560 Meters

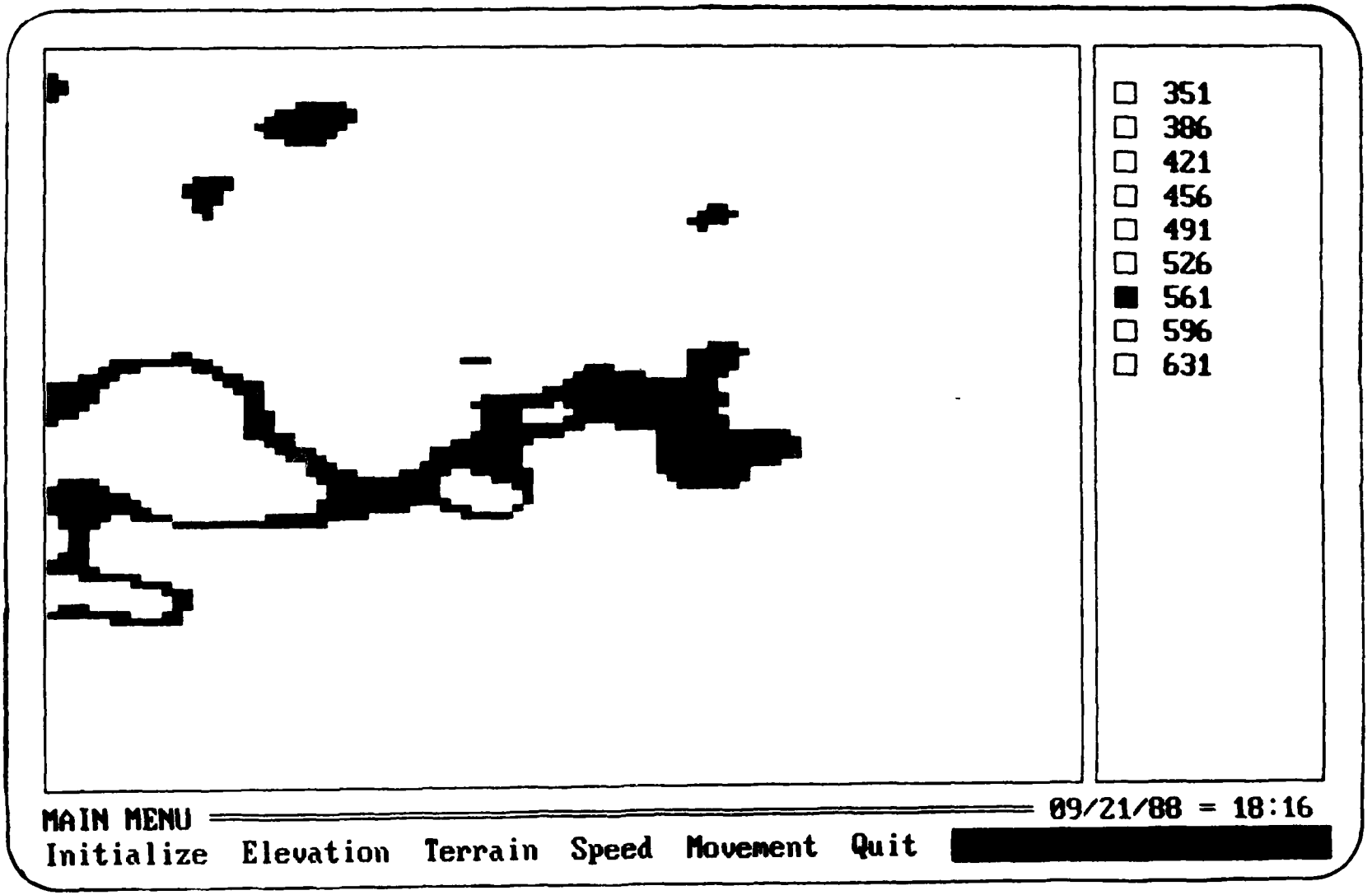


Figure 13. Elevation Map - 561 to 595 Meters

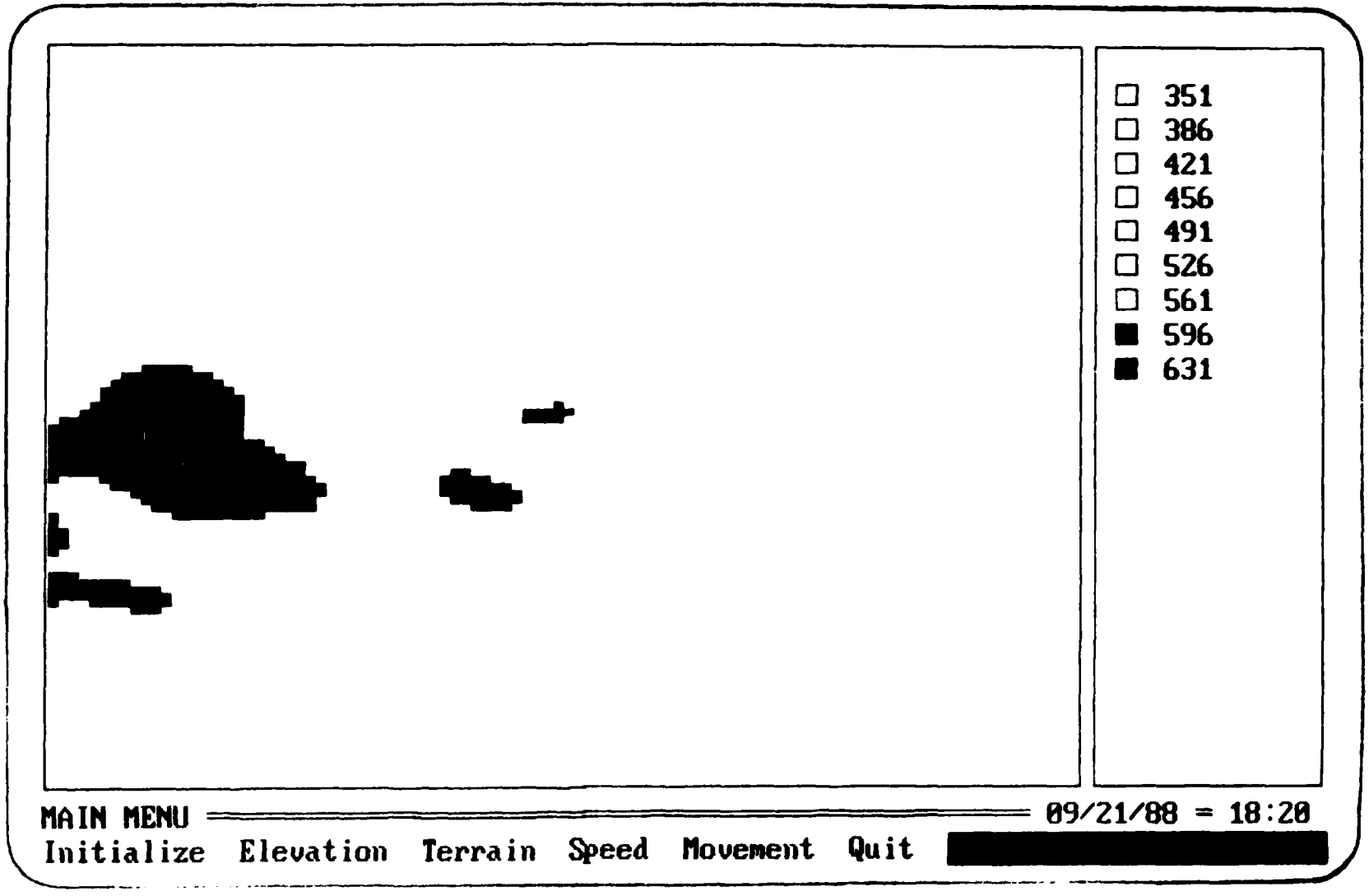


Figure 14. Elevation Map - 596 to 631 Meters

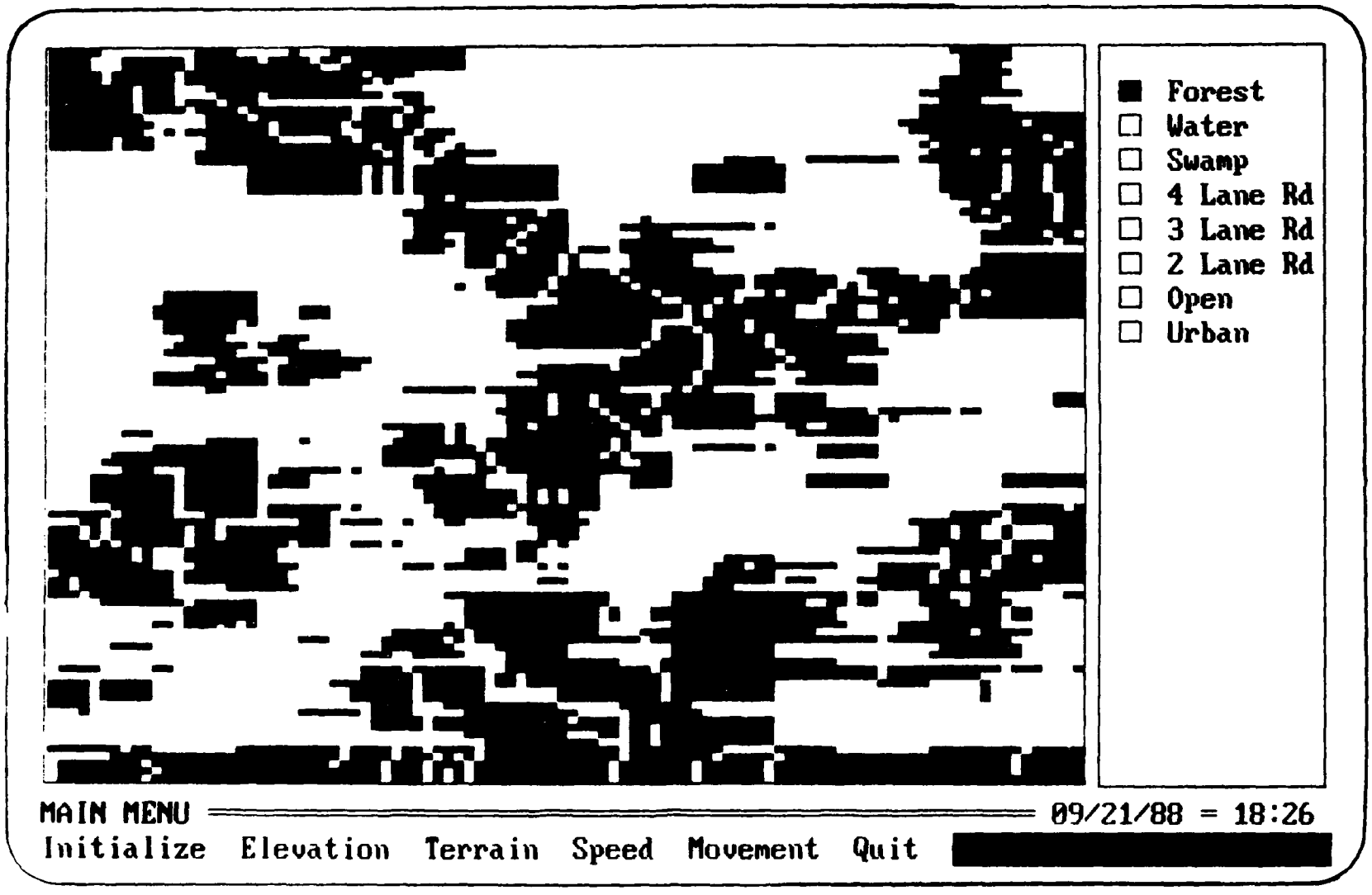


Figure 15. Terrain Map - Forest

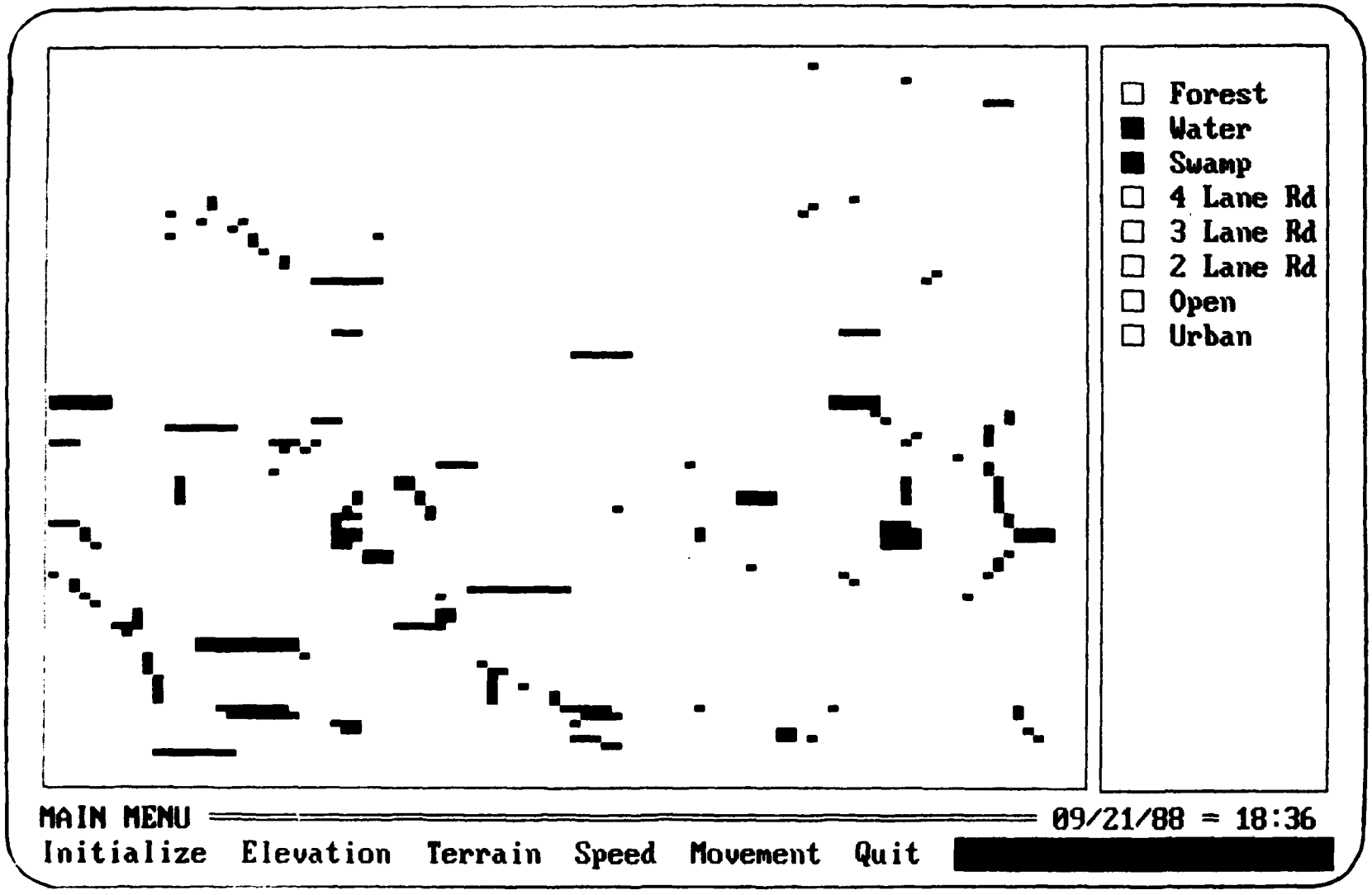


Figure 16. Terrain Map - Water and Swamp

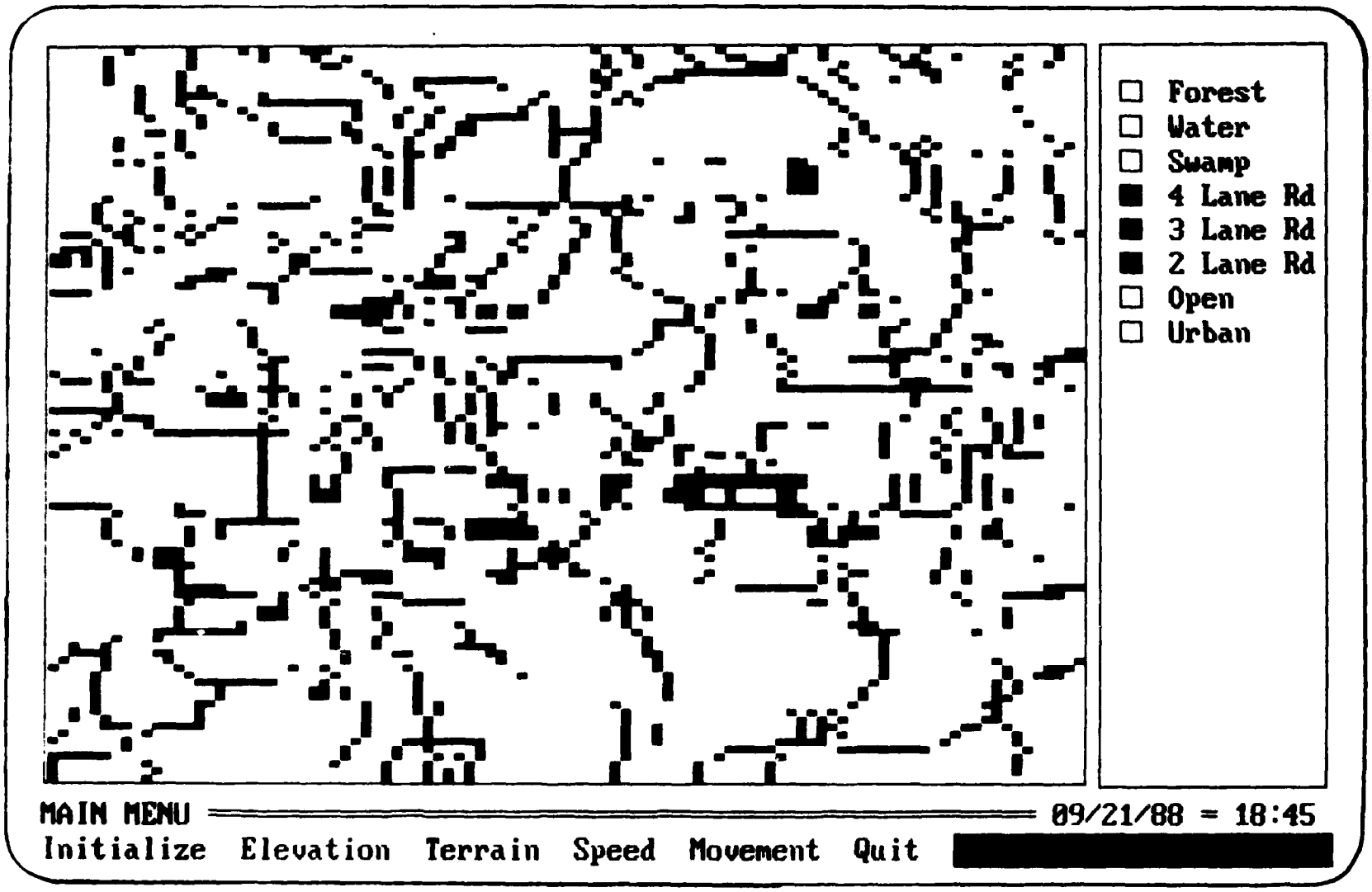


Figure 17. Terrain Map - Road Network

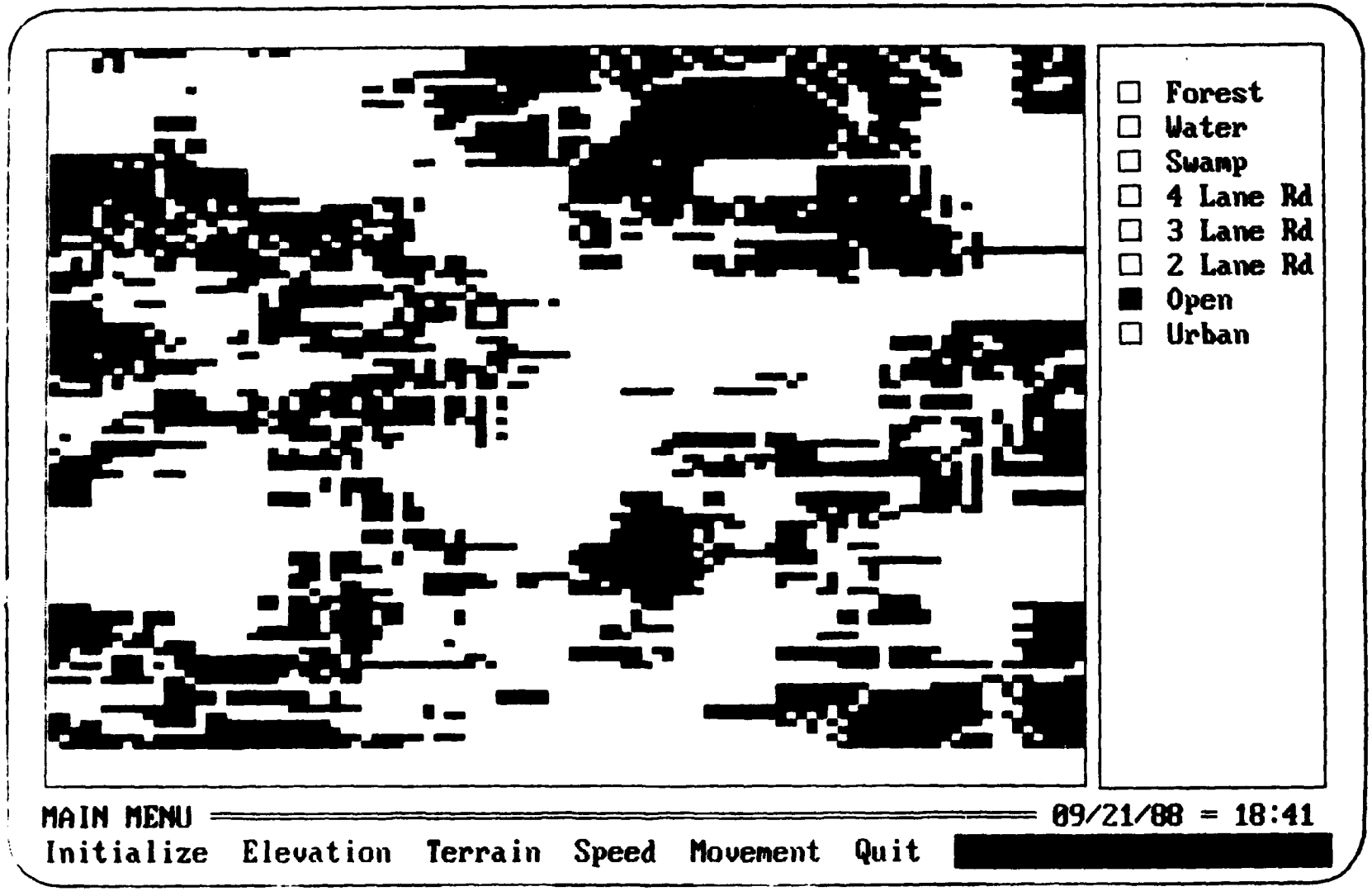


Figure 18. Terrain Map - Open

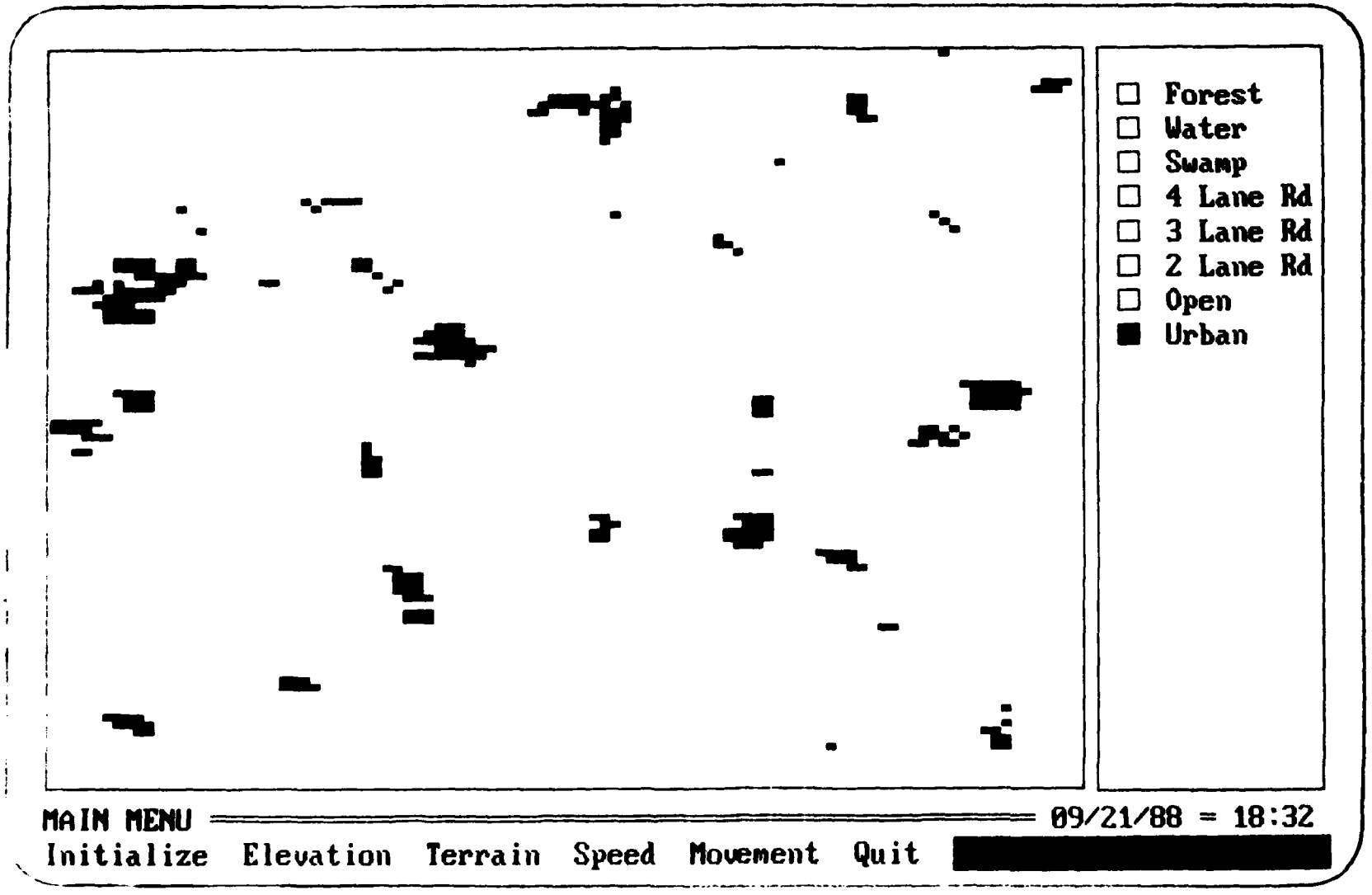


Figure 19. Terrain Map - Urban

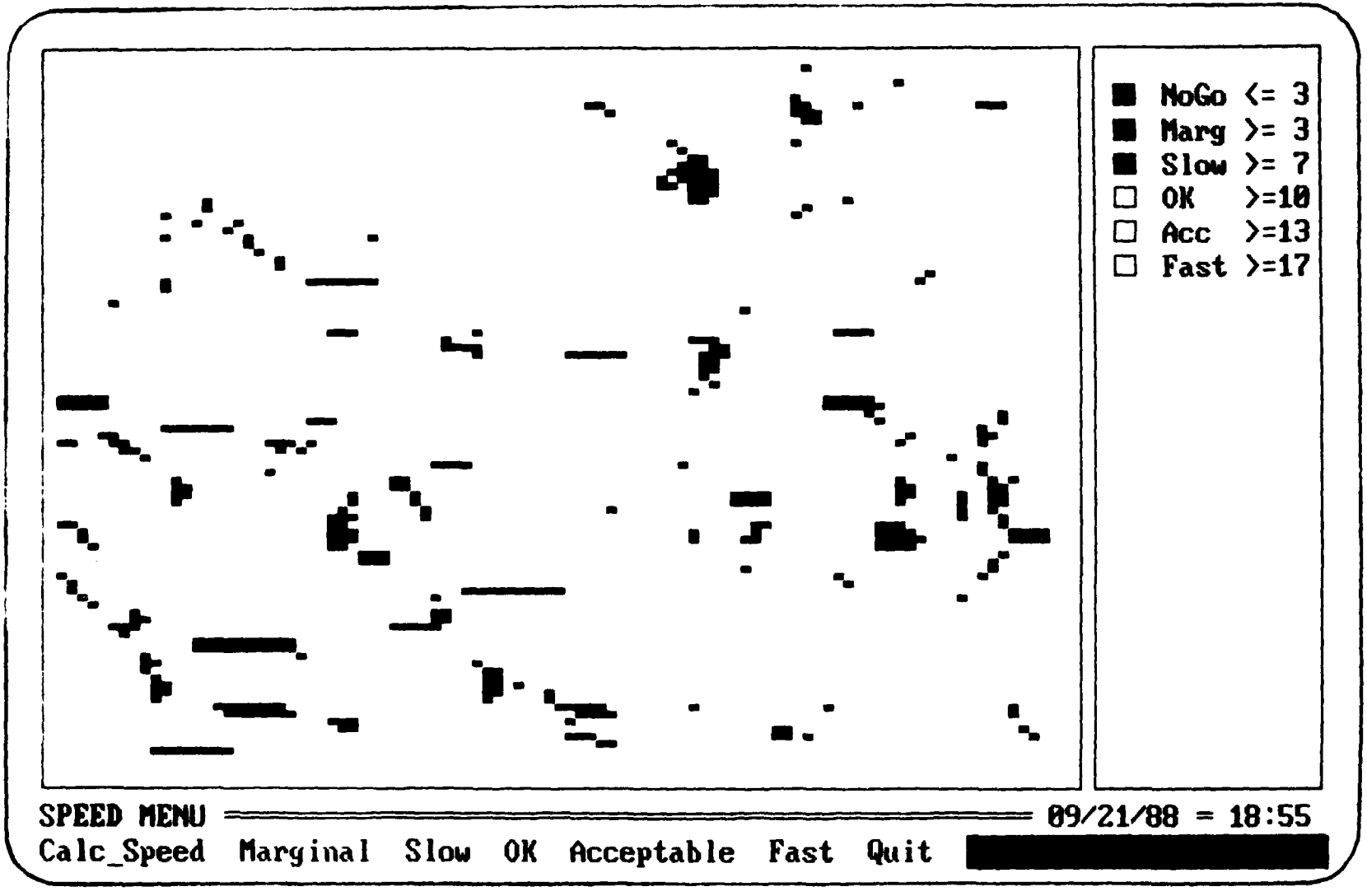


Figure 20. Speed Map - No Go, Marginal and Slow Points

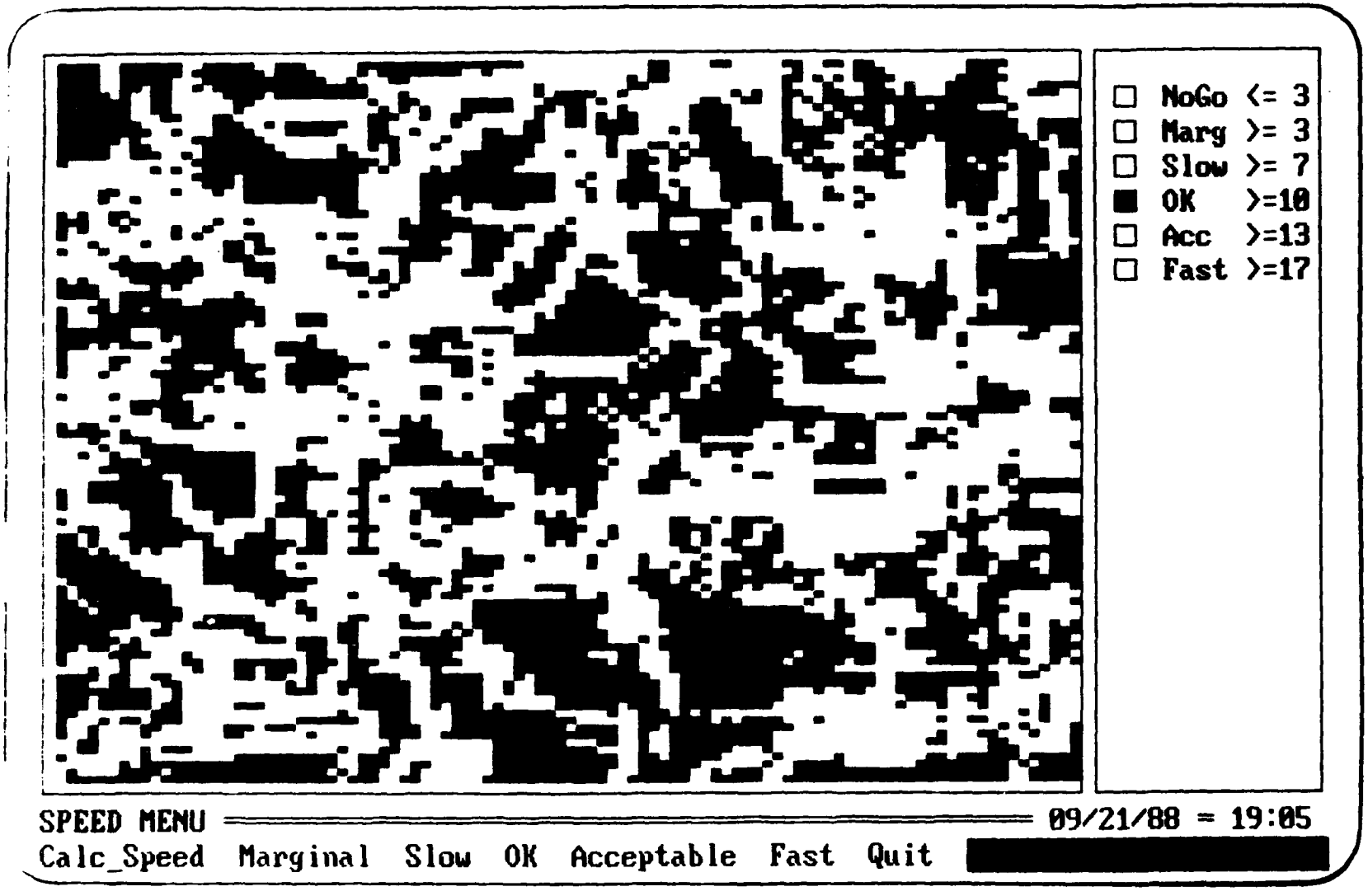


Figure 21. Speed Map - OK Points

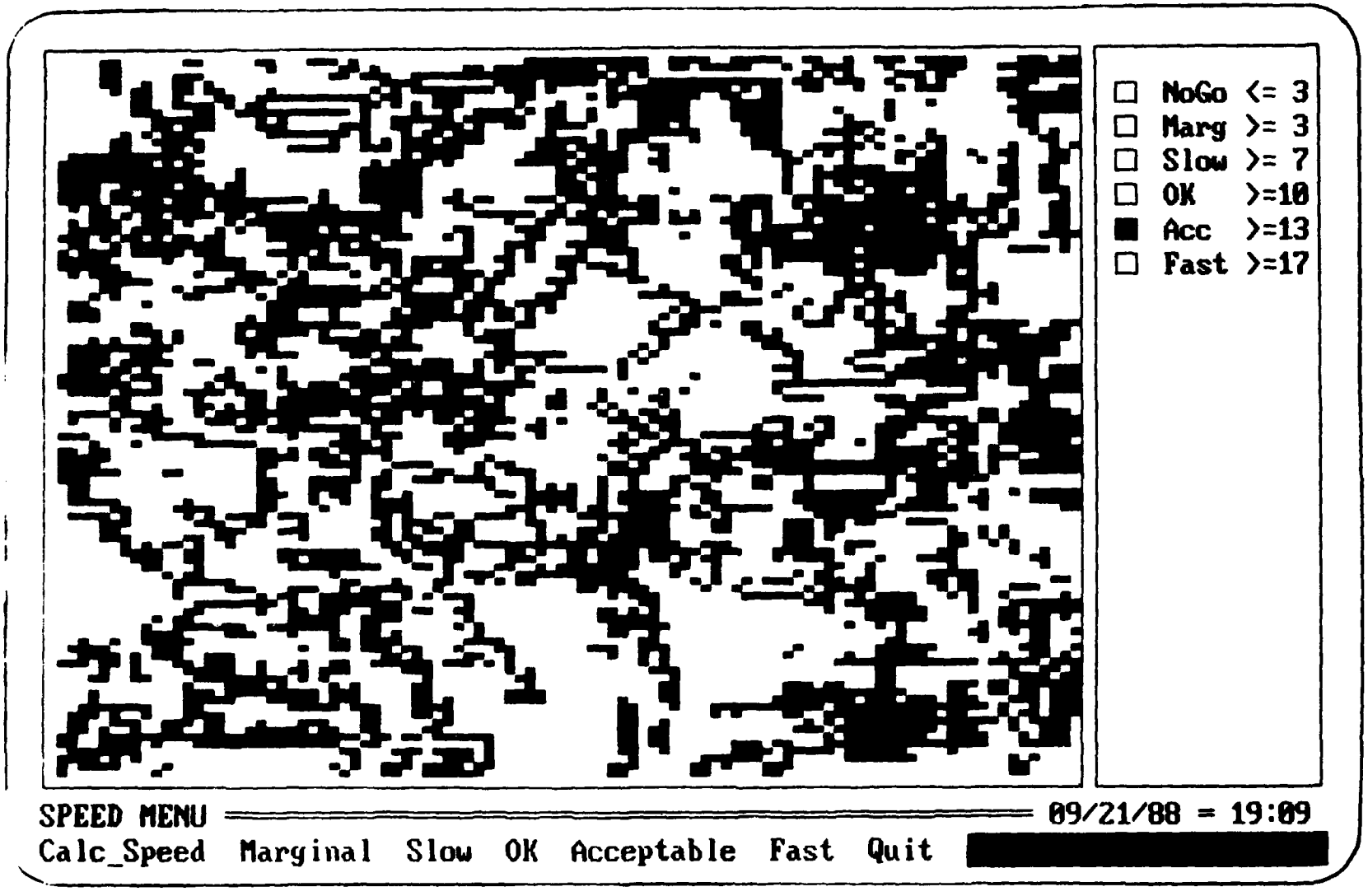


Figure 22. Speed Map Acceptable Points

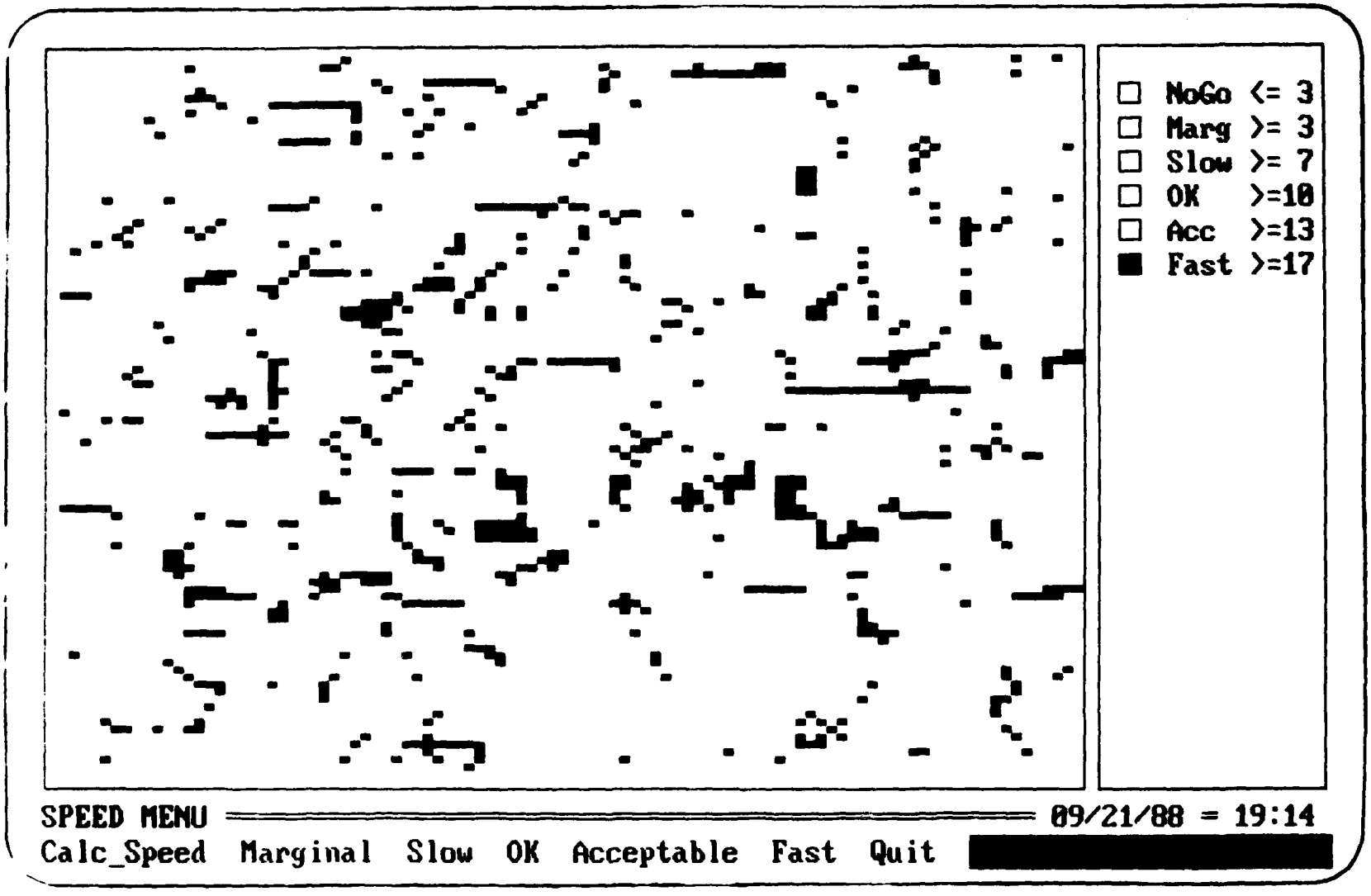


Figure 23. Speed Map - Fast Points

```
program speed_calculations;

type

{ Establish  pt_ar as a record of elevation and terrain
type for each data point }

elev_terr = record
  elev : integer;
  terr : byte;
end;
point_ar = array[1..105,1..105] of elev_terr;
point_ar_ptr = ^point_ar;

{ Establish speed_array as an array of each points speed
in the three directions of movement }

speed = array[1..105,1..105,1..3] of byte;
speed_pointer = ^speed;

var
  row, col, elevation, codes                          : integer;
  elev_file,terr_file,speed_file                      : text;
  pt_ar                                             :point_ar_ptr;
  speed_array                                        :speed_pointer;
  i, j, k                                             : integer;
  grad1_row, grad2_row, grad3_row, grad4_row          : integer;
  grad1_col, grad2_col, grad3_col, grad4_col          : integer;
  pt2_col, pt2_row                                    : integer;
  temp1, temp2, temp3, temp4, temp5, temp6            : integer;
  dist, slope, grad1, grad2, grad3, grad4             : real;
  max_slope, max_cross, terr_degr1, terr_degr2        : real;
  forest, water, swamp, urban, four_lane              : real;
  open,three_lane, two_lane                           : real;
  terr_degrade, slope_degrade, base_speed             : real;

begin

{ Get memory for the pointer arrays }
getmem(speed_array,sizeof(speed)+5000);
getmem(pt_ar,sizeof(point_ar)+5000);

{ Read in elevation file }
assign(elev_file,'east9_50.ele');
reset(elev_file);

while not eof(elev_file) do
begin
  for row := 100 downto 1 do
  for  col := 1 to 100 do
  begin
```

```
  read(elev_file,elevation);
  pt_ar^[col,row].elev := elevation;
  end;
end;
close(elev_file);

{ Read in terrain file }
assign(terr_file,'east9_50.ter');
reset(terr_file);
while not eof(terr_file) do
for  row := 1 to 100 do
for  col := 1 to 100 do
begin
  read(terr_file,codes);
  pt_ar^[row,col].terr := codes;
end;
close(terr_file);

{ Open speed file for output }
assign(speed_file,'east9_50.spd');
rewrite(speed_file);

{ Set the defaults }
  max_slope := 0.30;          max_cross := 0.30;
  forest    := 0.60;          urban     := 0.50;
  water     := 0.00;          swamp     := 0.00;
  four_lane := 0.95;          three_lane:= 0.90;
  two_lane  := 0.85;          open      := 0.70;
  base_speed := 20.0;

{ Determine each directions speed for each point and record
it }

for row := 3 to 98 do
begin

for col:= 99 downto 3 do
begin

    { For  each of the three directions of movement locate
    the gradient points and the next point }
    for k:= 1 to 3 do
    begin

    if k = 1 then
    begin
      dist := 141.4;
      grad1_col := col-1;        grad1_row := row+1;
      grad2_col := col+1;        grad2_row := row-1;
      grad3_col := col-2;        grad3_row := row;
      grad4_col := col;          grad4_row := row-2;
```

```
  pt2_col   := col-1;          pt2_row := row-1;
end;

if k = 2 then
begin
  dist  := 100.0;
  grad1_col := col;            grad1_row := row+1;
  grad2_col := col;            grad2_row := row-1;
  grad3_col := col-1;          grad3_row := row+1;
  grad4_col := col-1;          grad4_row := row-1;
  pt2_col   := col-1;          pt2_row := row;
end;

if k = 3 then
begin
  dist := 141.4;
  grad1_col := col+1;          grad1_row := row+1;
  grad2_col := col-1;          grad2_row := row-1;
  grad3_col := col;            grad3_row := row+2;
  grad4_col := col-2;          grad4_row := row;
  pt2_col   := col-1;          pt2_row := row+1;
end;

{ Calculate the gradients and the slope }
temp1 := pt_ar^[col,row].elev;
temp2 := pt_ar^[grad1_col,grad1_row].elev;
temp3 := pt_ar^[grad2_col,grad2_row].elev;
temp4 := pt_ar^[pt2_col,pt2_row].elev;
temp5 := pt_ar^[grad3_col,grad3_row].elev;
temp6 := pt_ar^[grad4_col,grad4_row].elev;

grad1 := abs(temp1 - temp2) / dist;
grad2 := abs(temp1 - temp3) / dist;
grad3 := abs(temp4 - temp5) / dist;
grad4 := abs(temp4 - temp6) / dist;
slope := abs(temp1 - temp4)/ dist;

{ Insure the gradients and slope do not exceed the max}

if ((grad1 >= max_cross) and (grad2 >= max_cross))
       or ((grad3  >= max_cross) and (grad4 >=
            max_cross))
       or (slope >= max_slope) then
         speed_array^[row,col,k] := 0

else
begin

{ Get the terrain degrade for each point }
case pt_ar^[col,row].terr of
 1 : terr_degr1 := forest;
```

```
        2 : terr_degr1 := urban;
        3 : terr_degr1 := swamp;
        4 : terr_degr1 := water;
        5 : terr_degr1 := open;
        6 : terr_degr1 := four_lane;
        7 : terr_degr1 := three_lane;
        8 : terr_degr1 := two_lane;
      end;

      case pt_ar^[pt2_col,pt2_row].terr of
        1 : terr_degr2 := forest;
        2 : terr_degr2 := urban;
        3 : terr_degr2 := swamp;
        4 : terr_degr2 := water;
        5 : terr_degr2 := open;
        6 : terr_degr2 := four_lane;
        7 : terr_degr2 := three_lane;
        8 : terr_degr2 := two_lane;
      end;

      { Insure both points are passable }
      if (terr_degr1 = 0.00) or (terr_degr2 = 0.00) then
         speed_array^[row,col,k] := 0

      else
      begin

      { Calculate the terrain and slope degrades }
      terr_degrade := (terr_degr1 + terr_degr2) / 2.0;
      slope_degrade := 1 - slope;

      { Set that points speed according to the following}
      temp1 := round(base_speed * terr_degrade *
               slope_degrade);
      speed_array^[row,col,k] := temp1;

      end;

     end;
    end;

    { Write each of the speeds to a file for future usage}
    writeln(speed_file,speed_array^[row,col,1],'  ',
                       speed_array^[row,col,2],'  ',
                       speed_array^[row,col,3]);
  end;
  end;

end.
```

```
program movement_paths;

 type

{ Establish speed_array as an array of each points speed in
  the three directions of movement }

 speed = array[1..105,1..105,1..3] of byte;
 speed_pointer = ^speed;


{ Establish the path_array as an array of the shortest time
  paths, including each step and the time at that step }

  ar1 = record
     row,col : byte;
     time    : integer;
     end;
  path = array[1..101,1..101] of ar1;
  path_ptr = ^path;


{ Establish node pointers to be used by AStar Search }

  node_ptr = ^node;
  node = record
     f : real;  {estimated cost from init to goal state}
     g : real;  {known cost of getting from init to state}
     xstate : byte; {current x cord of where you are}
     ystate : byte; {current y cord of where you are}
     parent : node_ptr;
     next : node_ptr; {link to the rest of the list}
  end;

 var

  open_list, closed_list                          : node_ptr;
  start_node,closed_node,best_node,child_node : node_ptr;
  cur_node,best_pred,prev_node                    : node_ptr;
  i,j,k                                           : integer;
  time_to_quit,replaced,dump_it                   : boolean;
  best_f, time, base_speed                        : real;
  speed_array                                      :
speed_pointer;
  mov_array                                       : path_ptr;
  move_file,speed_file                            : text;
  row, col                                        : integer;

 procedure astar(var row,col : integer);

    begin
```

```
{ Put the starting node on the open list }
new(start_node);
start_node^.xstate := col;
start_node^.ystate := row;
start_node^.f := 0.00;
start_node^.g := 0.00;
start_node^.parent := nil;
start_node^.next := nil;
open_list := start_node;

{ Set the closed list equal to nil }
closed_list := nil;

time_to_quit := false;

{ Repeat this loop until the stopping condition is met }


repeat

{ Loop  through the open list and find the node with the
 minimum f }


best_f := 1000.00;
cur_node := open_list;
prev_node := nil;
best_pred := nil;
while cur_node <> nil do
begin
  if cur_node^.f < best_f then
  begin
      best_f    := cur_node^.f;
      best_node := cur_node;
      best_pred := prev_node;
  end;
  prev_node := cur_node;
  cur_node  := cur_node^.next;
end;

{ Move best node from the open to the closed }

if best_pred = nil then
 open_list := best_node^.next
else
 best_pred^.next := best_node^.next;

best_node^.next := closed_list;
closed_list := best_node;
if best_node^.xstate = 1 then
   time_to_quit := true;
```

```
{If not time to quit generate children of the best node}

if time_to_quit = false then
begin

 for k:= 1 to 3 do
 begin
 if speed_array^[100-best_node^.ystate  ,100
                -best_node^.xstate , k] <> 0 then

 begin
    new(child_node);
    child_node^.parent := best_node;

    { Depending on the child the time and xstate is
 different}
    case k of
    1 : begin
        child_node^.ystate := best_node^.ystate - 1;
        time :=  ( 24  /
                  speed_array^[best_node^.ystate,
                  best_node^.xstate,k]) * 0.1414;
        end;
    2 : begin
        time := (24 /
                 speed_array^[best_node^.ystate,
                 best_node^.xstate,k])* 0.100;
        child_node^.ystate := best_node^.ystate;
        end;
    3 : begin
        time :=  (24 /
                 speed_array^[best_node^.ystate,
                 best_node^.xstate,k])* 0.1414;
        child_node^.ystate := best_node^.ystate + 1;
        end;
    end;

    { Establish childs other characteristics }

    child_node^.xstate := best_node^.xstate - 1;
    child_node^.g := best_node^.g + time;
    child_node^.f := child_node^.g + (child_node^.xstate
                     * 0.50);


    { If  child is already on the open list compare the
    f's and retain the best one }

    cur_node := open_list;
    replaced := false;
    dump_it  := false;
```

```
while cur_node <> nil do
begin
  if (cur_node^.xstate = child_node^.xstate) and
     (cur_node^.ystate = child_node^.ystate) then
     if cur_node^.f > child_node^.f then
     begin
        cur_node^.parent := child_node^.parent;
        cur_node^.f      := child_node^.f;
        cur_node^.g      := child_node^.g;
        replaced := true;
     end
     else
     begin
       dump_it := true;
     end;
  cur_node := cur_node^.next;
end;

{ If child is on the closed list and the f is better
  move the child from the closed to the open list }

cur_node := closed_list;
prev_node := nil;
if dump_it = false then
 while cur_node <> nil do
   begin
   if (cur_node^.xstate = child_node^.xstate) and
      (cur_node^.ystate = child_node^.ystate) then
     if cur_node^.f > child_node^.f then
     begin
        cur_node^.parent := child_node^.parent;
        cur_node^.f      := child_node^.f;
        cur_node^.g      := child_node^.g;
        replaced := true;
        if prev_node = nil then
          closed_list := cur_node^.next
        else
        begin
          prev_node^.next := cur_node^.next;
          cur_node^.next  := open_list;
          open_list := cur_node;
        end;
     end
     else
     begin;
       dump_it := true;
     end;
  cur_node := cur_node^.next;
  end;
```

```
        { If  the child is not on the open or closed list
       put it on the open list, otherwise dump it }

       if replaced or dump_it then
         dispose(child_node)
       else
       begin
         child_node^.next := open_list;
         open_list := child_node;
       end;

end; {end of if loop}
end; {end of k loop}

end; {end of time to quit}

until time_to_quit ; {end of repeat}


{ Once path is found write it to the move_file }

writeln(move_file,row);

cur_node := closed_list;

k := 1;
while cur_node <> nil do
with cur_node^ do
begin
  writeln(move_file,ystate,' ',xstate,' ',g:5:2);
  k := k + 1;
  cur_node := cur_node^.parent;
end;

{ Release temporary memory by clearing the open and
  closed lists.}

cur_node := open_list;
while cur_node <> nil do
begin
  dispose(cur_node);
  cur_node := cur_node^.next;
end;

cur_node := closed_list;
while cur_node <> nil do
begin
  dispose(cur_node);
  cur_node := cur_node^.next;
end;

end;
```

```
{ Main Program }

begin

{ Get memory for the pointer arrays }
getmem(speed_array,sizeof(speed)+5000);
getmem(mov_array,sizeof(path)+5000);


{ Read in speed file }
assign(speed_file,'east9_50.spd');
reset(speed_file);
for row:= 3 to 98 do
for col:= 99 downto 3 do
begin
readln(speed_file,speed_array^[row,col,1],
       speed_array^[row,col,2],speed_array^[row,col,3]);

end;



{ Open the move file for output }
assign(move_file,'east9_50.mov');
rewrite(move_file);

{ Make colomn 100 a feasible start colomn }

base_speed := 20.0;
for row:= 3 to 97 do
for k:= 1 to 3 do
begin
    speed_array^[row,98,k] := round(0.50 * base_speed);
end;

{ Set  impassable terrain on the north and south borders
 to force the algorithm to stay within the movement
 area }
for col:= 1 to 100 do
begin
  speed_array^[3,col,1] := 0;
  speed_array^[97,col,3] := 0;
end;

{ Run each start point through astar }

col := 98;
for row := 3 to 97 do
begin
    astar(row,col);
end;
```

```
{ Close the files }
close(speed_file);
close(move_file);

end.
```

## Bibliography


1. Bingham, Price T. "NATO Needs a New Air Interdiction Approach," Armed Forces Journal International: 98 - 112 (October 1986).

2. Brown, Jerome W. The Application of Single-Source Shortest Path Algorithms to an OJCS Contingency Planning Model and a Vehicle Routing Model. MS Thesis, Naval Postgraduate School, Monterey, CA, March 1987 (AD-A180118).

3. Choi, Seok C. Determination of Network Attributes From a High Resolution Terrain Data Base. MS Thesis, Naval Postgraduate School, Monterey, CA, September 1987 (AD-A186078).

4. Craig, Dean E. A Model for the Planning of Maneuver Unit and Engineer Asset Placement. MS Thesis, Naval Postgraduate School, Monterey, CA, September 1985 (AD-A159871).

5. Defense Mapping Agency, Product Specifications for Digital Landmass System (DLMS). Second Edition. Washington: Hydrographic/Topographic Center, February 1981.

6. Department of the Army. Countermobility. FM 5-102. Washington: HQ DA, 14 March 1985.

7. Department of the Army. Engineer Combat Operations. FM 5-100. Washington: HQ DA, 4 May 1984.

8. Department of the Army. Obstacle Planning Simulation, Version 1.1: Design and Performance Analysis. US Army Construction Engineering Research Laboratory, Champaign, IL, January 1985 (AD-A149468).

9. Department of the Army. Operations. FM 100-5. Washington: HQ DA, March 1986.

10. Dupuy, Trevor N., Numbers, Predictions and War. London: Macdonald and Jane's Publishers, Ltd., 1979.

11. Fletcher, Douglas L. Model for Avenue of Approach Generation and Planning Process for Ground Combat Forces. MS Thesis, Naval Postgraduate School, Monterey, CA, March 1986 (AD- A175318).

12. Hartman, James K. "Lecture Notes in High Resolution Combat Modeling". Unpublished Manuscript, Naval Postgraduate School, Monterey, CA, June 1985.

13. Hartman, James K. "Lecture Notes in Aggregated Combat Modeling." Unpublished Manuscript, Naval Postgraduate School, Monterey, CA, June 1985.

14. Hartman, James K. Parametric Terrain and Line of Sight Modelling in the STAR Combat Model, Technical Report, NPS55-79-018, Naval Postgraduate School, CA, Aug 1979.

15. Kazimer, Robert V. Combat Engineer Allocation Model. MS Thesis, Naval Postgraduate School, Monterey, CA, December 1984 (AD-A154182).

16. Kilmer, Robert. Using the Generalized Value System for Future State Decision Making. MS Thesis, Naval Postgraduate School, Monterey, CA, December 1984 (AD-A 168446).

17. Krupenevich, Thomas P. Network Representation for Combat Models. MS Thesis, Naval Postgraduate School, Monterey, CA, December 1984 (AD-A154068).

18. MacLaughlin, Joseph R. The Extension of Unit Allocation and Countermobility Planning Algorithms in the Airland Research Model. MS Thesis, Naval Postgraduate School, Monterey, CA, March 1986 (AD-A168393).

19. Morris, John W. "Combat Engineers - Mobility - History Airland Battle." Individual Essay, US Army War College, Carlisle Barracks, PA, March 1987 (AD-A180416).

20. Mustin, T. M. Optimal Allocation of Air Strikes for Interdiction of a Transportation Network. MS Thesis, Naval Postgraduate School, Monterey, CA, June 1967.

21. Nugent, R. O., The Optimal Allocation of Airstrikes Against a Transportation Network for an Exponential Damage Function. MS Thesis, NavaL Postgraduate School, Monterey, CA, October 1969.

22. Parry, Sam H. and Arthur L. Schoenstadt. Toward an Axiomatic Generalized Value System. Naval Postgraduate School, Monterey, CA, May 1986.

23. Rich, Elaine, Artificial Intelligence. McGraw-Hill, Inc., 1983.

24. Siferd, Jeff, Analyst, Aeronautical Systems Division, Avionics Lab. Personal interview. Air Force Systems Command, Wright Patterson, AFB, OH. 1 July 1988.

25. Slattery, Patrick J. A Structure for the Development of an Engineer Model. MS Thesis, Naval Postgraduate School, Monterey, CA, December 1980 (AD-A097290).

26. Wollmer, R. D., "Algorithms for Targeting Strikes in a Lines-of-Communication (LOC) Network," Operations Research, 18: vol 3, pg 497, (May-June 1980).

VITA

Captain Cynthia E. Staley was commissioned into the Army Corp of Engineers in August 1980 through the ROTC scholarship program at the University of South Florida, Tampa, Florida. Her civilian education includes a Bachelor of Science in Civil Engineering. Her military education includes the Engineer Officer Basic and Advanced Courses and the Combined Arms Services Staff School. Captain Staley has served in a variety of engineer duty positions in West Germany and CONUS. Her assignments included: Platoon Leader, Executive Officer, Assistant Brigade Operations Officer and Company Commander. Captain Staley entered the School of Engineering, Air Force Institute of Technology, in June 1987.

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution limited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>AFIT/GOR/ENS/88D-17 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>School of Engineering<br>6b. OFFICE SYMBOL (If applicable)<br>AFIT/ENG | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State, and ZIP Code)<br>Air Force Institute of Technology<br>Wright-Patterson AFB OH 45433-6583 | 7b. ADDRESS (City, State, and ZIP Code) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State, and ZIP Code) | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO. |
| 11. TITLE (Include Security Classification)<br>See Box 19 | |
| 12. PERSONAL AUTHOR(S)<br>Cynthia E. Staley, B.S., Cpt, USA | |
| 13a. TYPE OF REPORT<br>MS Thesis<br>13b. TIME COVERED<br>FROM _____ TO _____<br>14. DATE OF REPORT (Year, Month, Day)<br>1988 December<br>15. PAGE COUNT<br>78 | |
| 16. SUPPLEMENTARY NOTATION | |

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | |
| 32 | 06 | 3 | Combat Engineering, Countermobility, Obstacle Emplacement, Avenue of Approach Generation, Decision Support Tools. |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: A PROTOTYPE DECISION SUPPORT TOOL FOR ENGINEER COUNTERMOBILITY PLANNING

BS Przemieniecki 10 Jan. 89

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Bruce W. Morlan, Maj, USAF | 22b. TELEPHONE (Include Area Code)<br>(513) 255-3576<br>22c. OFFICE SYMBOL<br>AFIT/ENG |

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSFIED