NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DETECTION AND TRACKING BASED ON A DYNAMICAL HIERARCHICAL OCCUPANCY MAP IN AGENT-BASED SIMULATIONS

by

Dietmar Josef Teufel

September 2008

Thesis Advisor:	Christian Darken
Second Reader:	Kevin Squire

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.		

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Detection and Tracking Based on a Dynamical Hierarchical Occupancy Map in Agent-Based Simulations		**5. FUNDING NUMBERS**
6. AUTHOR(S) Dietmar Josef Teufel		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Agent-Based Models in military simulation need a model for detection and tracking other agents. It has been suggested that statistical models, such as occupancy maps or particle filters, can be used for that purpose. An occupancy map is one possibility for this task. The more volume of space, however, in a simulation, the more the computational demand of using occupancy maps grow and the more benefit could be obtained by the ability to switch to a coarser granularity in at least some parts of the volume.

Using both possible benefits of an occupancy map, fine granularity in tracking and detection where needed and less computational demand by switching to low granularity where possible, parts of the volume will be transferred to a new occupancy map on a higher hierarchal level with coarser granularity. Only the most interesting areas in the simulation have fine granularity.

The main contribution of this research will be to provide an improved algorithm and a prototype for using a hierarchy occupancy maps in agent-based simulations involving large volumes of simulated space.

14. SUBJECT TERMS tracking, detection, agents, occupancy map, simulation, probabilistic target tracking, hierarchical graph, abstract graph	**15. NUMBER OF PAGES** 83
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

DETECTION AND TRACKING BASED ON A DYNAMICAL HIERARCHICAL OCCUPANCY MAP IN AGENT-BASED SIMULATIONS

Dietmar Josef Teufel
Lieutenant, German Navy
Graduate-Engineer (FH), University of German Armed Forces Munich, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2008

Author: Dietmar Josef Teufel

Approved by: Christian Darken
Thesis Advisor

Kevin Squire
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Agent-Based Models in military simulation need a model for detection and tracking other agents. It has been suggested that statistical models, such as occupancy maps or particle filters, can be used for that purpose. An occupancy map is one possibility for this task. The more volume of space, however, in a simulation, the more the computational demand of using occupancy maps grow and the more benefit could be obtained by the ability to switch to a coarser granularity in at least some parts of the volume.

Using both possible benefits of an occupancy map, fine granularity in tracking and detection where needed and less computational demand by switching to low granularity where possible, parts of the volume will be transferred to a new occupancy map on a higher hierarchal level with coarser granularity. Only the most interesting areas in the simulation have fine granularity.

The main contribution of this research will be to provide an improved algorithm and a prototype for using a hierarchy occupancy maps in agent-based simulations involving large volumes of simulated space.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Chris Darken, for his guidance and expertise in this area of research. This helped me to finish my thesis work successfully. He motivated me during his classes to start a thesis with a topic in this research area of simulation. During the thesis process, he was patient with me when I had thesis difficulties. He gave me the right advice at the right time.

I would also like to thank all the Naval Postgraduate School professors, who lectured in the Department of Computer Science of the Naval Postgraduate School for their support and council during the study, and especially for answering any question if needed. Without the timely support of the readers this Thesis would never have been finished.

Finally, I would like to thank my classmates at the Computer Science department and at the Moves Institute for their support through the study and companionship.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM

For years, modeling human-like behavior has been one of the aspects of research in the fields of artificial intelligence and military simulations. The more an agent behaves -- in an Agent-Based Simulation -- like a human being, the more realistic the simulation. Like a human in the real-world, the agent will make similar decisions and errors. To reach this aim, many methods have been developed over the years. These were more or less successful.

Tracking and detection are some of the human-like behaviors. They play an important role in both military simulation and in real-world military. One of the most important pieces of information in military operations and in military simulation is the opponent's position. If the information is correct, there is a higher probability of success; on the other hand, if the information is wrong, there will be less success. Key to a military simulation is following the knowledge of the environment. One resulting variance of such a simulation is the uncertain reasoning about the opponent's position.

To ensure realistic behavior and to achieve valid results, agent-based models in military simulation have models for detection and tracking other agents. It has been suggested that statistical models, such as occupancy maps or particle filters, can be used for that purpose. There is likelihood that the agent will track his opponent on the wrong position. This is because the agent does not really know about the opponent's position. Like real persons, he estimates his position on the location with the highest probability. This possible failure gives an agent a more human-like behavior and, therefore, makes the outcome of such a simulation more realistic.

One method for tracking is an occupancy map. It is a directed graph. Each node of the graph has a geographic position; each edge represents the

possibility to move from one node to the other. Different methods are particle filters, which represent the position of a target as a set of possibilities. These are called particles.

Occupancy maps are relatively simple to develop and to use for probabilistic tracking. Because of this, they are widely used in the area of simulation, i.e., computer games and robotics.

However, with the increasing power of computers in the last years, there is also a need of the military simulation community about more complex simulation. The simulation area, also the map the simulation should work increases over the last few years. Occupancy maps work very exact and fast in small sized simulation, therefore there are often used in such simulation, because of there simplicity

The occupancy map, however, has a disadvantage in large-sized simulation: the larger the volume of space in a simulation, the more the computational demand of using occupancy maps grows. Also, more benefit could be obtained by the ability to switch to a coarser granularity in at least some parts of the volume. In the outer, or in less interesting regions of the simulation, the probability of each node will be calculated with the same cost as in the area near the search agent. This is a waste of resources: not only is it unrealistic to calculate the location of an opponent with such precision, but it is not necessary. If a simulation is huge and has a wide area, most of the computations about the probability of an opponent would be less useful. They would only serve to bind the computer resources for tracking.

On the other hand, if the granularity spread is caused by an increasing area, there is a different disadvantage: the computational cost remains constant; further, the granularity in the area close to the searching agent will be wide. This could lead the searching agent to imprecise tracking and to artificial behavior in the close area.

There is a need for a solution which combines the advantages of the occupancy map -- especially the possible high precision in the area close to the search agent and the simplicity of building and using this kind of probability tracking. Additionally, the improvement should avoid the disadvantage of high cost of calculation probabilities in both the simulation's outer area or in uninteresting locations.

B.   RESEARCH

Using both possible benefits of an occupancy map -- fine granularity in tracking and detection (where needed) and less computational demand by switching to low granularity (where possible), parts of the volume will be transferred to a new occupancy map on a higher hierarchal level with coarser granularity. Only the most interesting areas in the simulation have fine granularity. This dynamic behavior itself has computational costs. The focus of the thesis will be to develop an algorithm to use these hierarchal occupancy maps and to reduce the cost of the original fine granularity occupancy map.

The scope of this thesis includes retaining the advantages of occupancy maps, with a fine granularity in detection and tracking, in large-scale simulation. To achieve this, the occupancy maps are divided into different levels of abstraction, i.e., a hierarchy. The building of different levels in this hierarchy during runtime of the simulation (dynamic building) will itself incur computational cost. In the past, this principle was used for path-finding. In this thesis, this principle is used to develop the algorithm for such a hierarchical graph for probabilistic tracking.

After the basics of the hierarchy occupancy map were developed, the development of a prototype followed. With an experimental prototype, the concept of the hierarchy occupancy map will be proven. Initial testing of the prototype will be done, to ensure the software-prototype of the hierarchy

3

occupancy map fulfill the requirements as described during the modeling process and behaves as expected. This will test the algorithm for building and working with a hierarchical occupancy.

Currently, there is no prototype or environment to proof or test the concept of using hierarchy maps for target detection and tracking. The main contribution of this thesis is to deliver a prototype for the proof of concept of hierarchical occupancy maps in agent-based simulation.

The experimental prototype should be implemented in a modular way. In the future, the prototype could be used beyond the scope of this thesis. Additionally, different experiments could be done in future work. The prototype will be useful to test and following improve the algorithm for hierarchal occupancy maps in future research.

C.   THESIS ORDER

The thesis is organized as follows:

- **Chapter II, Previous Work in Target Detection and Abstract Graphs** describes various techniques for targeting with probabilty reasoning. Advantages and disadvantages of the different methods will be discussed. Additionally, the technique of abstract graphs will be declared and shown how it is used for path-finding.

- **Chapter III, Model and Algorithm** describes the development of the algorithm of the hierarchical graphs. The model will be described and explanation as to why it is useful in the simulation. The principles of a hierarchy graph will be declared. The focus will be on the static and dynamic behavior of such a graph in a simulation for detection and tracking.

- **Chapter IV, Design of the Prototype** describes the purpose of the developed prototype for the agent-based simulation. The development of the overall Software-Architecture of the prototype

4

will be declared. Special focus will be placed on the different classes and modules of the software. They will also be described. Additionally, the viewpoint from the discrete event simulation will be discovered and the function of the event graphs will be declared. Problems and discoveries during implementation are described.

- **Chapter V, Analyses of the Dynamical Hierarchal Occupancy Map** describes the initial tests of the prototype. Analysis of the visual appearance of the prototype will be made. Additionally, some metrics and quantitative results will be discovered from the simulation.

- **Chapter VI, Conclusion and Further Work** summarizes the contribution this thesis made to this topic. From the basics of the thesis, it will also give suggestions about further work.

THIS PAGE INTENTIONALLY LEFT BLANK

II.   PREVIOUS WORK IN TARGET DETECTION AND ABSTRACT GRAPHS

Military simulations which are agent-based should have a technique for target detection and target tracking. In previous simulations -- especially in the military or game development communities -- they often worked around this problem. In the past, developers often gave complete environment knowledge to all agents in the simulation. Target detection and tracking was more and less simplified or randomized to find the target agent or not. With time, there was a demand for more realistic simulation and tracking and detection methods. Over the last few years, several methods were developed and improved. Many of them work with probability distribution.

With this method, a searcher tracks the target on a map with a distribution probability of the location of the targets. There are two sub fields: one is the collaborate tracking of targets; the second is path-finding. In both fields, however, the agents need a valid representation of the target.

In following chapter, several models for probability distribution and target detection will be discussed and compared. Also the basics of an abstract graph will be discussed.

A.   OCCUPANCY MAP

Occupancy maps are discrete representations of a probability at a specific position in a map over time. The essence of an occupancy map is the projection of a grid in a specific environment. The grid itself is a directed graph. It is adapted to this environment as displayed in the figure below.
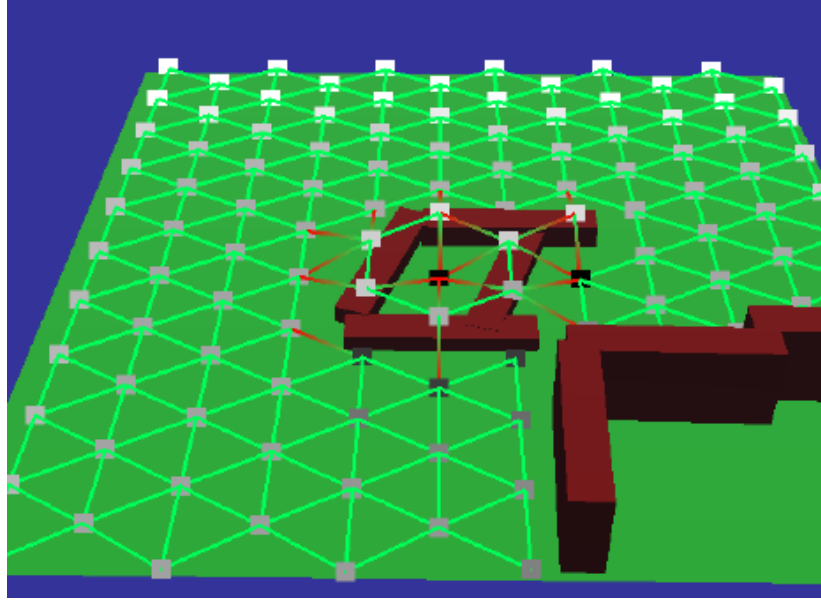
Figure 1.   Occupancy Map

Each node of the graph has a specific coordinate in the map. The edge between two nodes means that an agent can reach the second node over the edge from the first node. Following the edge represents the possibility of moving. If there is no edge between two nodes, there is no way to move from the one node to the other in a direct way. Each node is static, will not move over time, and stores a value p_A, which represents the probability a specific target is placed at this node [1]. The sum of all values of the nodes in the grid is calculated as follows:

$$p_{sum} = \sum_n p_A$$

The expected probability p for the target is at the specific node and is, then, calculated as follows:

$$p = \frac{p_A}{p_{SUM}}$$

8

The edges, which connect two nodes, have a specific parameter λ_{AB} which is a metric of the probability a target moves from node A to node B. This measurement can depend on the environment. If λ_{AB} is small, it is difficult to move from node A to node B. The probability that a target moves from node A to B will be less. If a specific node, however, has a beneficial position, the probability is higher. There are many reasons for setting the parameter λ_{AB} to a specific value. If there is no edge between node A and node B, the parameter λ_{AB} is zero. For each node,

$$\sum_A \lambda_{AB} \leq 1$$

The move-cull process is necessary for frequently updating the occupancy map and the distribution of the probability over the grid. If we define $p_A(n)$ as the value at a specific node after n time steps, the probability the target is at a specific node will be calculated as follows:

$$p_A(n+1) = p_A(n) - \sum_B \lambda_{AB} p_A(n) + \sum_C \lambda_{CA} p_C(n)$$

The probability that the target is at a specific point is calculated as sum to the probability which is distributed to neighbor nodes and the probabilities which come from the neighbor nodes.

The cull part of the process reduces the probability of nodes that are visible during the search process of the searcher. If the target is visible, the probability of the node, which at the position of the target will be set to one, the probability of the other nodes will be set to zero. If the target is not visible, the probability of all visible nodes will be set to zero. The probability at the other nodes represents the probability that the target is at this specific position.

The advantages of occupancy maps are that they are easy to generate and the distribution of probability is easy to calculate. If the move-cull process and the parameter between the edges are synchronized with the speed of the

target, the results of tracking are valid. Therefore, occupancy maps are widely used in the area of simulation, game development, and in map navigation of robots as a real world application.

Occupancy maps, however, have some disadvantages. One disadvantage is the possibility of magic movement of the probability between two nodes that are not visible to the searcher -- although the edge is visible. The culling process in the occupancy map cares only about the nodes and not about the edges. For example, if both nodes are not visible to the searcher, but the edge between nodes is, there is a possibility that probability distribution will be over this edge. This could lead to wrong solutions of the probability distribution over the nodes. Synchronizing the parameters within the edges and with the speed of the target could be difficult. If the synchronizing is invalid, the wave front of high probabilities could move too fast or too slow compared to the target agent. This could increase the error rate of tracking and make the model of tracking invalid. Additionally, the number of nodes and the computational cost of the tracking algorithm depend on the number of nodes and the size of the grid. If the map increases and there is only one single target to track, the computational cost increases.

B.   PARTICLE FILTER

A different approach for localization and tracking is particle filters. It is a particle-based state estimation technique. Particles represent a position of a target as a finite set of possibilities [1]. Each particle has a weight, which is proportional to the probability that the target is at this time at the specific position of the particle. Contrary to occupancy maps, the particle has free movement over the whole map. There is no grid, which leads the particle to specific positions. If the particle is visible from the searcher, it will be deleted. If after some time an area of the map is not visible to the searcher, a set of new particles on a specific or random place is generated. The new set of particles moves with random direction (360˚) and speed.

Similar to occupancy maps, a move-cull process is necessary. During culling, all visible particles will be deleted. If the target is visible, there will be no generation of new particles. The move-process is the initialization and moving of new particles. The parameter of each particle, such as velocity, is chosen at random. Particles can only move to positions which are reachable. The speed of movement could be selected at random or it could be determined by the environment. If the environment is difficult to move through, it could decrease the speed of the particle. If the environment is easy to move through, the particle's speed could increase.
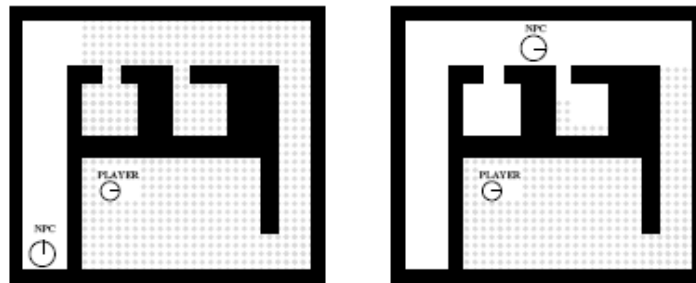
Figure 2.   Particle Filter (from [2])

Figure 2 demonstrates the particle filter in the special case that the particles are moving on a fine grid. The NPC is a searcher-agent. It searches for a player who is hiding in one of three rooms. Every position, which is and was visible, is white; thus, the particles disappeared. This means that the probability that the player hides at this specific place is zero. If the searcher-agent searches in the different rooms, more and more particles are deleted. This follows the probability that the player is at a place where remaining particle increase.

Also, with occupancy maps, there has to be a collision test of the single particles to avoid "magic movement." If a particle hits a wall or barrier, it has to be considered in the movement of the particle itself. The particle has to be eliminated or its direction is changed.
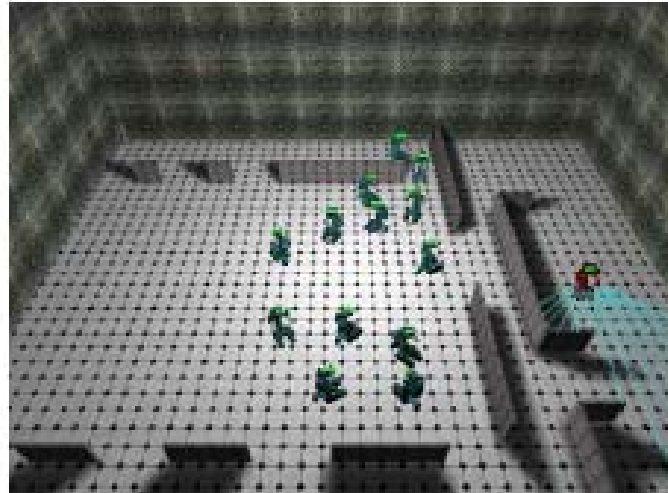
Figure 3.   Game Level with Particle Filters (from [2])

When a visible target moves out of sight as displayed in the figure above, a specific number of particles are placed on a specific position. This initial position could be extrapolated from the position and speed of the target at the time it disappeared. Now all particles move over time through the map with the move-process. It is frequently proofed if the particle is visible and will be deleted with the cull process.
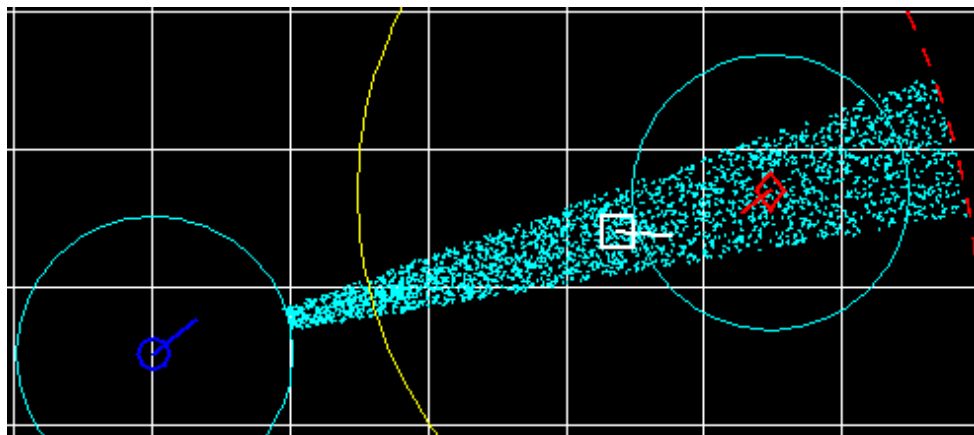
Figure 4.   Particle Filter for Tracking (from [3])

Additionally, a good demonstration of particle filter is the tracking of opponents with real sensors. Particle filters are not only used in the field of simulation and game development, but also in the field of tracking real data from sensors. They are also used in making an estimate of the position of the target if the tracking is interrupted, such as in Radar Technique [4]. Figure 4 is a simulation of tracking a target with different sensors in Borovies Master Thesis [4]. At this moment, the searcher uses a bearing-only sensor with a specific tolerance. The green dots are the particles which are more or less uniformly distributed in the area in which the target could be. The white dot is the position which is most likely for the target. It also provides the most likely speed and direction of the target.

Particle filters have some advantage over occupancy maps: particles could move exactly between the minimum and maximum speed of the target; thus, the distribution of the particles is more realistic than in an occupancy map. This is because there is not static grid to limit the movement. Additionally, the particles can be spotted and deleted as they move. This avoids magic teleportation between two hiding places which sometimes occurs in occupancy maps. The number of particles is completely independent from the size of the level in a simulation or game. With an occupancy map with increasing level size, this research increased the number of nodes and noted the computational cost or the decrease of the granularity level. Particles are independent of that. The granularity depends on the number of particles generated and this is independent from the size of the level or map.

Particle filters do come with disadvantages: the random choice of movement is sometimes not realistic. In a specific environment, a target does not behave randomly; rather, it will have preferred speed and direction. In such circumstances, the initialization of the particles with random direction and speed will lead to a less accurate tracking or the implementation of this behavior lead to additionally computational cost. Additionally, particles need a collision test. If a particle reaches an area which cannot be reached by the hiding agent, the

13

particle has to be deleted or moved to another direction. The algorithm for detecting this collision, and the following changing of the particle, has an additional computational cost compared to other methods of tracking. The computational expense of particle filters is directly proportional to the number of particles. If the number of particle is too small, there is danger of no reasonable tracking. The tracking of multiple targets needs multiple sets of different particles. Every target needs its own set of particles to get its own probability distribution.

C.   SIMULACRA

Simulacrum is a hybrid model of tracking and target detection. It combines the advantages of particle filters and occupancy maps. It was introduced by Darken and Anderegg in their article [1]. It is based on particle filters which gave a good visual representation of how the search agent tracked the target agent. A simulacrum tries to simplify the model for particle filter and also to deliver a more realistic tracking behavior. For example, the tracking of targets should sometimes be assumed by the searching agent and be more independent from the environment where the target is moving. This is because the search-agent does not know the environment and, therefore, cannot make assumptions about the speed of the hiding-agent. For the representation, the probability particles are used, but instead of moving free on the map, they now move on the edges of an underlying grid -- or navigation path -- similar to an occupancy map.
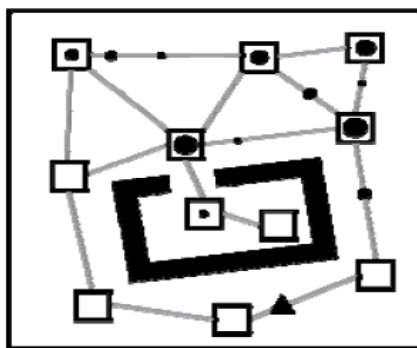
Figure 5.   Simulacra (after [1])

14

Figure 5 shows a typical example of simulacra. The probability of the target is represented by the size of each node in the graph. The wandering of the probability, or dynamic behavior, is displayed as dots moving on the edges from one node to the neighbor's node.

The particles of simulacra have a position x_n and a weight p_n. A behavior state is optional. This behavior state, however, makes the model more complicated. On the other hand, it could lead to a more realistic behavior of the target. To support this, consider that the target hides with higher probability at the nodes. The search-agent knows these preferred nodes.

The sum of p_n is

$$p_{sum} = \sum p_n$$

N is the number of current particles, or simulacra, and N* is the number of desired simulacra. During the dynamic behavior of the probability distribution, if N<N* there will be a split of simulacra. The simplest approach is to split the simulacra into two new simulacra with half probability p_n. During the move process, splits occur until the number of simulacra N=N* is achieved. Following on every node, the probability will split. In models that are more complex, there could also be a probability not to split. It may also not follow a specific edge to the next node. This is because the cost to reach this node is too high.

The culling process is the same as with particle filters. If a particle becomes visible, the particle will be deleted and its value p_n will be subtracted from p. Additionally, N has to be subtracted by one. This is because in the tracking model now only one particle is lost.

Simulacrum combines the particle filter with the advantages of a navigation path. The movement of the particles is not random anymore. This avoids the necessity for collision detections of particles and following reduces the

computational cost compared to basic particle filters. It also solves the problem of magic movement: if a particle is visible for the searcher wandering between two nodes, the particle will be deleted.

D. ABSTRACT GRAPHS

Search algorithm, such as A*, normally uses the full representation of the underlying graph. It also tries to find an optimum solution -- or path -- from one specific node to the target node. In the field of computer simulations, it is sometimes not necessary to find the optimal solution. This is because it needs too many resources and computational power. A near optimal, but very fast, solution will be more beneficial. A solution for path-finding, developed by N. Sturtevant and M. Buro [5], is shown in their paper about abstract graphs.

An abstract graph is a reduction of a full-state graph where each node represents one or more states in the lower level graph. An edge exists between the nodes in the higher level. This occurs if, in the lower level, an operator can reach one of the other nodes which is represented by the next higher level node. A node in a higher level represents the information of a set of one or more nodes of the lower level.
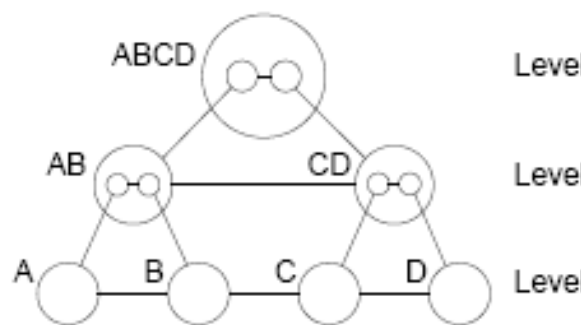
Figure 6.   Building Levels (from [5])

16

An abstract graph with three levels is displayed in Figure 6. The full representations of the graph on the first level are the nodes A to D. For the path-finding algorithm, it is necessary that only nodes which are connected go to the next level. As the nodes A and B and, on the other side C and D, are connected, they will be represented by a common node AB and CS in the next level. The only information necessary for the next level is whether the previous nodes are connected outside with another node. If it is connected, there will be an edge between the representation of the node and the target node outside. When this procedure is repeated with the next level, the result is one single node for this graph. This means there is -- at minimum -- one path between all nodes of the graph.

To use an abstract graph for path-finding, the selected start node and target node in the next level are used until both are represented by the same node. If there is no common representation, there is no path between the two nodes. The path could be built from the common node.
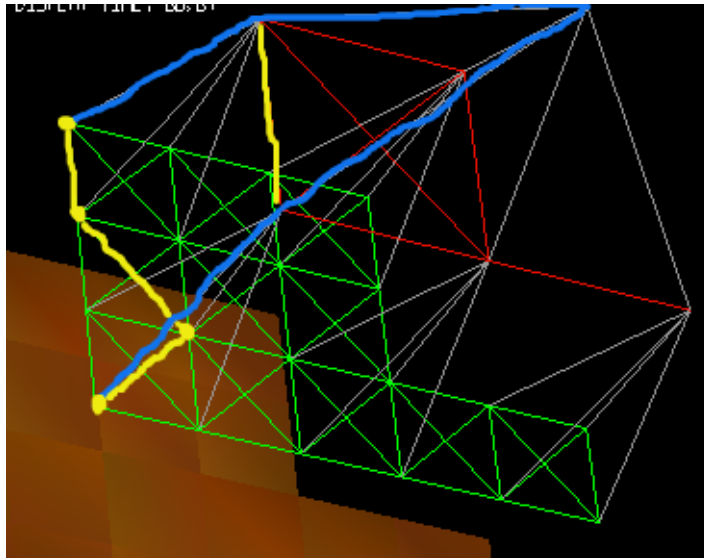
Figure 7.   Quick Path with Abstract Graph

In Figure 7, the path-building process is displayed. In every lower level, the algorithm must find a path between the representation of the starting node and the target node. If the quick path algorithm reaches the full representation of the lowest level, it is the demanded path from the starting node to the target node. The path may only be suboptimal; however, for every level to find the path, the algorithm needs O(log n).

The cost for this is that the path is only suboptimal. Additionally, the cost of building the abstract graph itself is suboptimal. N. Sturtevant and M. Buro [5] proofed in their paper that the cost for the initial building of the abstract graph is O(n).

The idea behind the abstract graph is to use, in every higher level, a representation of the information of the lower level. In this thesis and, especially in the following chapter, this principle is used as a representation, not for path-finding, but for a probability distribution to improve occupancy maps.

III. MODEL AND ALGORITHM

A. DEVELOPING THE GRAPH

In this chapter, the environment of the simulation will be described. It shows how the occupancy map will work within this environment. The algorithm and method to build a hierarchy occupancy map will be declared. The focus is on the dynamic behavior of the graph during runtime of the simulation.

1. Environment

A military agent-based simulation of combat situations is sometimes comparable to games developed by the computer game industry. There are similar problems to solve; therefore, there are similar solutions for the problems -- especially in the research field of Artificial Intelligence. The reason is that both, especially real-time games, are a kind of simulation. The National Research Council recommends techniques which were originally developed for games in military simulations [6]. A common major problem is target tracking and detecting. In both, there is an ever-increasing power of computers and a demand for larger simulations and larger game levels. If the level of games and the displayed area of simulation become huge, new difficulties in the field for target tracking and localization appear.
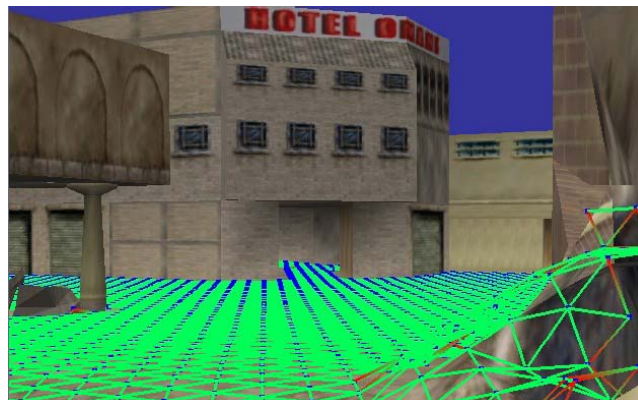

Figure 8.   Typical Occupancy Map in a Game Level

The one-level occupancy map and the other techniques, as described in the previous chapter, show that particle filters and simulacra have different types of difficulties when dealing with this increasing complexity. Simulation and gaming, however, sometimes deal with these kinds of problems differently, too.

When a realistic solution has a computational cost that is too high, the game developer often cheats. With too realistic behavior, the game player will not be entertained as well (Bererton, 2004 [2]). For example, the game software tells the search agent where the human opponent is. Thus, the programmer can avoid developing a complex target tracking and searching algorithm. Further, the human player does not have to wait a long time for the agent to find his position. For the game player, such an artificial intelligence of the opponent agent looks like a realistic behavior, but, in the background of the agent, it is not realistic behavior.

In military simulation, this is not a valid solution. To reach valid and more realistic results from an agent-based model, the agents in the model must make human-like decisions. A part of this decision is the search for opponents. As described in previous chapters, the occupancy map is a good method to reach this human-like behavior in military agent-based simulations. Therefore, it is used widely. The node represents the probability that the searched agent is at this specific location. Therefore, for a huge agent-based simulation, the number of nodes will increase. As the area of the simulated terrain increases, there are several new problems. If the granularity of the probability tracking is constant, the number of nodes will increase. Also, the time and memory for tracking and hiding agent with these probability nodes will increase. The developer of the simulation can make the decision to decrease the number of nodes or to reduce the computational cost of the simulation. It will, however, decrease the granularity of the grid and, with that, the precision of tracking opponent's agents. Therefore, we

loose quality in the tracking algorithm of the simulation and the realist behavior of the agents. This could lead to less human and realistic behaviors of the agents in the military simulation.

Simulation Area	Constant Granularity 10m x 10m	Constant Node 100
100m x 100m	100 Nodes	10m x 10m
200m x 200m	400 Nodes	20m x 20m
500m x 500m	2500 Nodes	50m x 50m
1 km x 1 km	10,000 Nodes	100m x 100m
5 km x 5 km	250,000 Nodes	500m x 500m
10 km x 10 km	1,000,000 Nodes	1000m x 1000m

Table 1   Occupancy Map Growing Table

A valid solution would be to track precisely near the agent and to track less precisely further away. Then the granularity of the grid, or underlying map, will increase near the agent and decrease with more distance from the agent. The search agent needs the most precise occupancy map for this position nearest him. This makes possible, during the run of the simulation, a realistic search in his neighborhood. The more distance from the searcher a node has, the less it has to be precise. A precision target and detection over a far distance is not necessary and not realistic. Following the granularity of an occupancy map could decrease the furthest nodes of the map away from the searching agent.

To find the target agent, the searching agent will search over the occupancy map at the areas which have the highest probability. This means that such an occupancy map is not static: the areas of detailed granularity must move with the searching agent over the simulation map. There is a need for dynamic

behavior of such an occupancy map. When the searcher moves, there must be an information exchange between the different levels of this map.

To fulfill the previous written requirements of an occupancy map in an agent-based simulation with a huge simulation area, graph theory is used to build an occupancy map with different levels of details. To reach this aim, a graph with different layers is used. Such a graph already exists in Sturtevant and Buro's paper: using map abstraction for path-finding [5]. In this thesis, the technique is used and modified to develop an occupancy map with different granularity on different levels. In this paper, it is called hierarchy graph or hierarchy map.

2.   Hierarchy Graph

The occupancy map is used as the foundation to develop the basic hierarchy graph. Each node of the basic graph represents one specific area; the edge represents the cost to move from one area to the next. The node also holds the probability that a target is in the specified area.
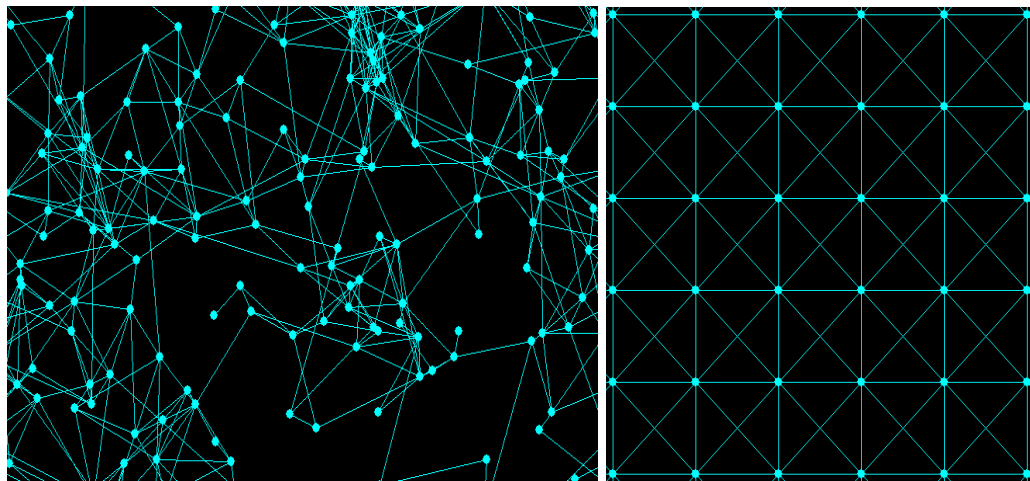
Figure 9.   Two Basic Graphs of an Occupancy Map

Hierarchy graph is a finite sequence of graph's *(G1..Gn)* where Gi=(Vi,Ei) and there exists V_{i+1} which is an abstraction of Vi. The different graphs are called levels of the hierarchy graph. Development of the levels and, therefore, of the

whole hierarchy graph should be done during the initialization of the simulation. The cost of the building the hierarchy graph is not considered for the runtime of the simulation itself.

From the basic graph, the first step is to develop one or more of the upper levels of the hierarchy graph. The method is to put some nodes of the previous level or hierarchy together as one set of nodes and let them represent a node from the next level.
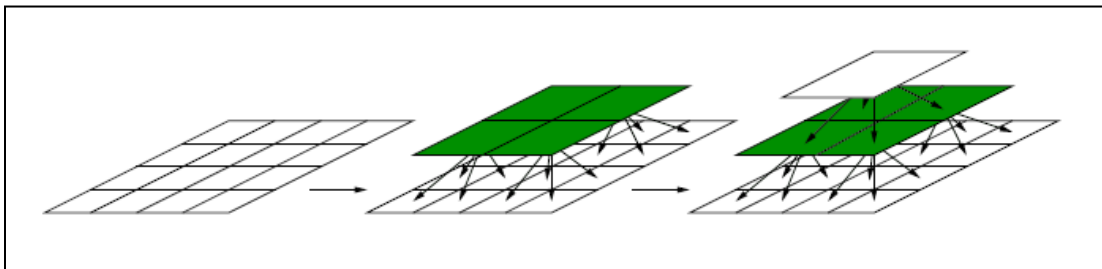
Figure 10.   Abstracting a Graph in Levels (from [5])

Theoretically, this procedure can be repeated until there is only one node to represent the whole grid of the simulation. In practice, achieving one node depends on how huge the simulation area is and how detailed the granularity should be. The output is a hierarchical graph, which has different levels of sub graphs. Every level ensures a reduction of the number.

The reduction metrics is chosen based upon the number of nodes. These nodes are represented by one node in the next hierarchy. During the simulation's initiation, a huge reduction has the advantage of less levels of hierarchy. It could, however, increase the demand for computing the dynamic behavior of the hierarchy graph. This is because there are more nodes that the probability has to calculate during the reduction to one node (or expanding to the previous level). In addition, the granularity of tracking the target decreases faster during the runtime of the simulation.

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│              *Algorithm Developing Next Hierarchy*                    │
│                                                                       │
│ *procedure NextHirachie (G, n)*                    **the overall procedure** │
│                                                                       │
│ **Input:**        G=(V,E) is a directed graph; with vertex v ∈ V,     │
│                   n is number of nodes in cliques,                    │
│ **Output**:       new hierarchical graph G'; with v' ∈V'              │
│                   and u∈ U as edges between G and G'                   │
│                                                                       │
│ Create G'                                       **create an empty Graph g'** │
│ While some v not connected to a node in G'and n >1                    │
│       For all v∈ V not connected                                      │
│               procedure BuildNClique (G, G' v, n) **calls for new clique** │
│       n= n-1                                                          │
│                                                                       │
│ For all v∈ V not connected,                                           │
│       put single node to next clique                                  │
│                                                                       │
└─────────────────────────────────────────────────────────────────────┘
```

Figure 11.   Algorithm Developing Next Hierarchy

In the algorithm for developing the next level, the input value n is the number of maximum nodes in a clique. The term clique in the algorithm is a set of nodes which are fully connected. The first cliques are searched with n nodes in the basic or previous graph. If there is no remaining n-clique, the next step is to search for a clique with n-1 nodes. This process is repeated until there are only single nodes which are not in a clique. These single nodes can be added to an existing clique or can be represented by a single node in the next level of the graph.
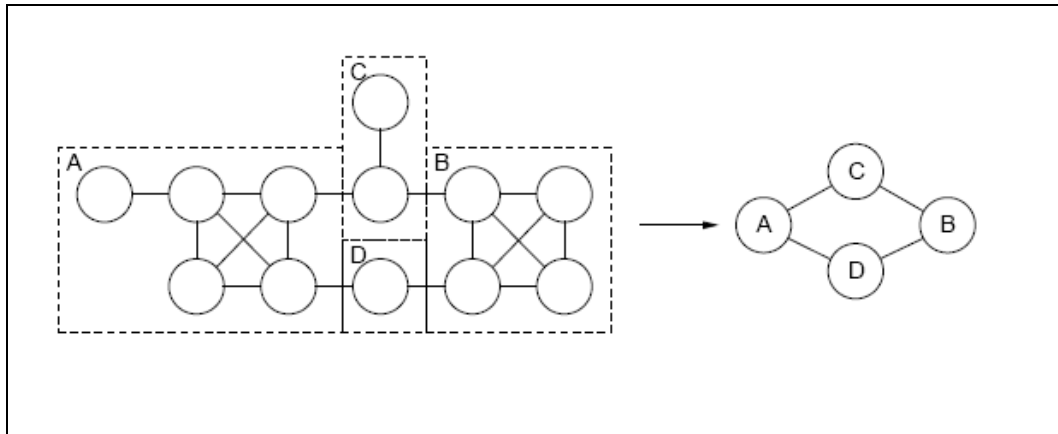
24

Figure 12.   Building Cliques

According to Corman [7], a clique is a subset of vertices of the graph G=(V,E) -- each pair of which is connected by an edge in E. A clique is a complete sub graph of a graph. This means that every node in the clique could be reached with one step of each node in the clique. Each clique will be represented in the next level of the abstract graph as node. The principle was used by Sturtevant and Buro for building levels of hierarchy for path-finding [5].

For every node in the basic graph, the algorithm proofs first, i.e., the node could reach n-1 neighbour nodes with one step. If this test is positive, the node is put on a list of possible clique nodes. To build a clique, each node (of the possible clique-list) must be able to reach its neighbour node with one step and, also, the neighbour node must be able to reach this node in a directed graph. Both must be proven. Further, the algorithm must prove that if one of the nodes -- or the neighbour's node -- is not already a member of a clique, it will connect to the graph of the next level. If the algorithm found n nodes which fulfil the requirement, the algorithm will build and connect -- with a separate procedure -- the hierarchy node of this clique.

Figure 13.   Algorithm Building a Clique

The algorithm for building the hierarchy node takes the information from the underlying clique to the hierarchy node. This represents the clique in the next level and connects this node to hierarchy graph G'. The new hierarchy node is created and connected to the right level of the graph with edges. For that, it is mandatory to prove if the edge of a node of the underlying clique goes inside (travels to another node of the same clique) or outside (travels to another clique). In the next level graph, the node is only connected with the edges, which are not internal edges of the clique. The new hierarchy node represents a more large area. The internal edges of the clique are inside the new represented area and are not considered in the next level of wider granularity.

Algorithm Building a Hierachy Node

procedure BuildHierarchieNode (G, G', L)

Input: G=(V,E) is a directed graph; with vertex v ∈ V
            G'=(V',E') is the hierarchy graph
            L = list of nodes of the clique (w)
Output: w ∈ L connected to next hierarchy graph G'

Create new node w' as part of G'   ** the new hierarchy node**
for all w ∈ L ∪ {v}
        connect w to w'   **connect Node to the hierarchy graph**
for all u which are connected to w
        if u not ∈ L   **the edges between the levels**
          if u not connected to G'
                connect u to w'   **connect clique with the hierarchy node
          else connect u' to w'
clear L

Figure 14.   Algorithm Building a Hierarchy Node

If all nodes are from the lower level and are connected with this algorithm to the higher level, the procedure for one level is finished. The whole algorithm now could be repeated to build the next level. The new developed level would be used as the new basic graph. Then, from the previous hierarchy nodes, new cliques would. be built for the next level. To build a hierarchy graph, the algorithm could be used in a recursive way.

B.   DYNAMIC BEHAVIOR

1.   Dynamic Behavior of the Graph

The dynamic behavior is newly introduced here for the hierarchy graph. It should ensure that only the nodes on different levels of the graph are considered. This is necessary for the distribution of the probability in an occupancy map. Normally, the nodes near the search agent are on the lowest level. Those further away and outside at a specific range are on the higher level. If in simulation there

are more than two levels with increasing distance from the search agent, the active node will be on a higher level. If the agent moves for the search during runtime of the simulation, nodes of the upper hierarchy appear in the distance should be put down to a lower hierarchy. On the other hand, if cliques of the lower hierarchy are completely out in the distance; it is not effective to consider this set of nodes in detail. This changing of the active nodes (better called dynamic behavior of the hierarchy graph) is the important function of the graph to reduce the active nodes and, also, the calculation of probability on a specific point during the runtime of the simulation.

As previous described, each node in an upper hierarchy holds the information of a whole clique in the lower hierarchy. During the process of movement of the searcher agent, the clique must give the information to the hierarchy node. The upper hierarchy node must be switched to active. This means it will be considered in the culling and distribution processes of probability. In addition, the clique in the previous level must be set to passive. This means it is not considered during culling and distributing the probability.

At the start of the simulation, the position of the searching agent on one node had to be set. The agent achieved the lowest hierarchy beginning in this area and, more outside, to the upper hierarchy. This is part of the initial process of the simulation, but the same algorithm can be used during runtime.

The algorithm to place the information of a node from a higher to a lower level -- or hierarchy -- is called *PopDown*. The algorithm must first test if an upper hierarchy node is in a specific distance to put it down to the next level. If this test is positive, all edges of the upper hierarchy node are disconnected from the upper hierarchy graph. This means that the edges and the nodes are set to passive, i.e., they will not be considered during the culling and distributing of probabilities. The neighbour nodes of the previous active node will be tested and, if necessary, put down to the clique node.

Algorithm PopDown

Procedure PopDown(G,G',A)

Input: G=(V,E) is a directed graph of lower hierarchy with vertex $v \in V$
                  G'=(V',E') is the hierarchy graph
                                  A = agent
Output: graph G'

For all $v' \in V'$
       If v' is in range:
              Disconnect all edges of v' in G'
              Add all children v of v' to G'
                    For every child v:
                            p(v) = p(v')/number of children of v
                            Connect all edges of v to the neighbours

---

Figure 15.   Algorithm PopDown Node

The edges leaving from the new clique will connect to the neighbour's clique or to nodes of the upper hierarchy. The edges of the neighbours' cliques are already on the lower level and must set to active. The edges of the node from the lower level which connects to the higher level or, the opposite, from the edge of the node of the higher level to the lower level, are temporarily at cross level edges. This connects both levels.
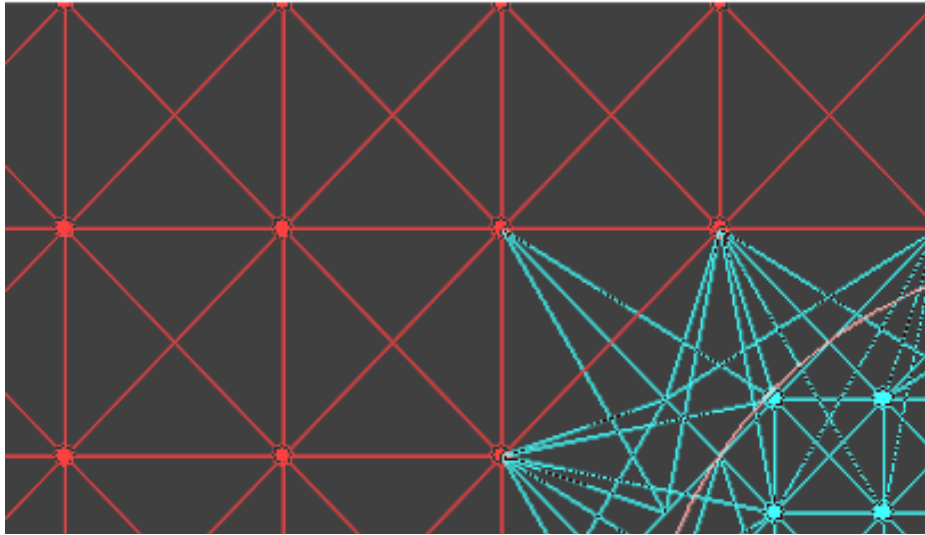
Figure 16.   Transition between Levels of Hierarchy Graph

This should ensure that, in the transition area between two levels, the probability could be distributed between the two levels. Figure 9 shows such a transition area between two levels. This transition area will be move with the searcher and, therefore, with the border of the distance which was chosen as the connection for the levels.

When the detailed nodes are set to active, the last step in the algorithm is to distribute the probability that the target is in the upper node in the detailed map. A simple method for this is to only share the probability to the number of nodes in a clique as

$$p(v) = \frac{p(v')}{\#nodes\_in\_Clique}$$

This simple formula, however, does not ensure terrain or special points will lead to a higher probability or that the searched agent is in one of the clique's nodes.

<div style="border:1px solid black">

Algorithm PopUp

Procedure PopUp(G,G',A)

Input: G=(V,E) is a directed graph of lower hierarchy with vertex v ∈ V
      G'=(V',E') is the hierarchy graph
      A = agent
Output: graph G'

For all v ∈ V
    If v is out range of agent
        And if all v with same clique is out range of agent
            Add parent v' to G'
            Connect all edges from v' to G'
            p(v') = 0
            For all v in clique
                p(v') **+=** sum p(v)   **calculate probability**
                Disconnect all edges E to GU
                Delete VLow from GU

</div>

Figure 17.   Algorithm PopUp Node

If a clique gets out of the chosen distance of the searcher agent, the algorithm must be used for the opposite way. This algorithm is called *Procedure PopUp* (Figure 17). First proof: if the inactive node of the upper hierarchy is out of range. If this proof is positive, it proofs if all member nodes of the clique in the detailed map are out of range. The next step is to set the node in the upper hierarchy active and to set the nodes in the clique passive. Distributing the probability will now occur only on the upper level node. This node connects or, better, sets all edges active with upper hierarchy. It also proofs the neighbour nodes which are still passive. The edges to these nodes will be temporary to the nodes of the lower hierarchy clique. It is the same principle of the transition edges which were described previously.

The last step is to sum up the probability of the lower nodes and to set it in the upper hierarchy node.

$$p(v') = \sum p(v_{clique})$$

2. Culling

As considered in the previous chapter, a node represents the probability of an opponent to stay at this location. The occupancy map or, better, the probability of the nodes must be updated with a move-cull process. This is different from the previous dynamic behavior of the map or graph. There is no change in the behavior of the graph itself -- only the probability of the nodes is changed. If there is a specific time or if it is in simulation with a specific time rate, the target must be checked for visible range.

If the node is visible and the agent is not on this node, the probability of the node is set to p(v) = 0. If the agent is visible, the nearest node has the probability p(v) = 1. For all other nodes -- visible or not -- the probability is set to zero p(v)=0.

<div style="border:1px solid">

Algorithm Cull

Procedure Cull (G',A, T)

Input: G'=(V',E') is the hierarchy graph
          A = agent searcher
          T=target
Output: graph G'

If T is visible for A:
      Nearest node p(v)= 1
      All other nodes p(v) = 0
Else
      For all active vertices v of G'
          If node is visible for A
             p(v) = 0

</div>

Figure 18.   Algorithm for Cull

Over time, the sum of all probabilities will decrease. Thus, the sum will always be smaller than 1.

$$p_{SUM} = \sum p(v)$$

$$p_{SUM} \leq 1$$

To calculate the probability at a specific time of a specific node follows

$$p(node) = \frac{p(v)}{p_{SUM}}$$

It is important to note that this is a more theoretic value -- all p(v) always have the same relation to each other. So the p(v) with the highest value also has the highest value of *p(node).*

33

3.   Distribution of Probability

The distribution of the probability on the occupancy map is basically the same as ordinary occupancy and hierarchical maps. In other publications (for example, in Darkens and Andereggs' paper [1]), distributing the probability is also called move-process. The difference between ordinary or hierarchy map is that, in the hierarchy map, some nodes are inactive. The nodes in a clique and the probability of the position of an agent will be displayed over the hierarchy node. Only the nodes which are near the searching agent will be distributed to the lower lever. The hierarchy node is inactive. Switching the node active or inactive was done during the dynamic behavior of the map (described in the previous chapters).

Following the algorithm of distributing probabilities between an ordinary map and an occupancy map is not different. In a directed graph, probability is going out and probability is going into a node. In each round or time stamp, all values of the nodes must be copied with a temporary value called p_{old}. The new $p(v)$ is the sum of the new incoming probability and the outgoing probability to the next nodes. Each edge from one node to another has a cost. That is the likelihood an agent takes this way if he was previously in the node. With a higher cost, the probability to take a specific edge to the next node is less likely.

Algorithm Distribute Probability

Procedure DistributeProbability (G')

Input: G'=(V, E) is the hierarchy graph
Output: Graph G'

     For every active vertex v of G'
          Copy p(v) in $p_{old}(v)$
          For every edge E outgoing
               Get target node v'
               $p(v') = p(v') + (1/\text{cost of E})*p_{old}(v)$
               $p(v) = p(v) - (1/\text{cost of E})*p_{old}(v)$

Figure 19.   Algorithm Distribute Probability

It is important to synchronize the move-process with the cull-process. If culling is done before moving, it could happen that the node the agent was recently p(v) is one and will be immediately culled to zero. It will not have the chance to distribute the probability. For that reason, the move-process should normally occur before the cull-process.

The speed or velocity that the probability is distributed depends on two factors. First, the frequency of the culling-move-process: during each move-process, the probability will be distributed over the edges to the next neighbor. If the move-process is culled frequently, the speed of the distribution will increase. Next parameter is the cost of the edges: if there is a high cost, only less parts of the probability will be distributed to the next node. If there is low cost, a higher portion will be distributed. During implementation of the model, there must be synchronization between both parameters. This ensures simulation of the speed of the hiding agent.

If the agent is not visible for the searcher for a long period during a cull process, the overall p_{sum} will decrease over time. After a long time, this p(v) could become very small. It could, in a large simulation, lead to the problem that the

values are too small to make a reasonable search process. Therefore, there must be an observer process which will refresh the values of p(v).

IV.   DESIGN OF THE PROTOTYPE

In this chapter of the thesis, there is description of the development of the prototype of the hierarchy occupancy map. The simulation library Simkit was used to build the prototype. The implementation was done in the programming language JAVA.

A.   PURPOSE OF THE PROTOTYPE

The prototype will show how the hierarchical occupancy map works. Additionally, it will be used as simulation to deliver data for future analysis of the hierarchical occupancy map.

To get all parameters from the simulation, listener classes will be added. The parameters for calculating the results of the simulation will be implemented in the Simkit-related classes. Also, the alphanumeric results of the simulation will be displayed in a separate and movable JAVA frame.

First, the prototype has to prove that the theoretical concept and algorithm developed in the previous chapter will work. Additionally, the performance of the prototype and the designed map will be proofed. For analysis to be performed with the delivered data, the map and the data should be displayed over the Graphical User Interface.

B.   ARCHITECTURE

1.   Modules and Classes

Modularization will be used to build the software for the prototype. During the development process, classes and packages will be developed from the modules of the architecture. Overall, the prototype is built in a layered architecture concept [9]. The Graphic User Interface (GUI) is the top layer. It should ensure that the underlying concept of the hierarchy occupancy map is displayed and that the user can see the alphanumeric values of the simulation. It is the interface between the user and the simulation. The second layer is the

discrete event simulation itself. It is the core of the prototype in which the algorithm for the dynamic and static behavior is implemented. The concept of discrete event simulation with event graph is used to simulate the dynamic behavior of the occupancy map.
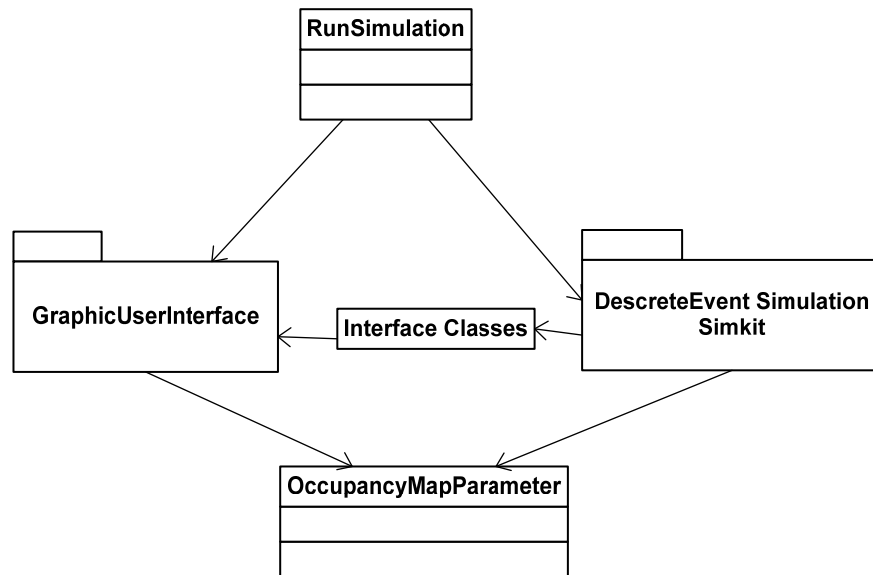

Figure 20.   Software Architecture Prototype

Figure 20 displays the architecture of the prototype. The two modules GraphicUserInterface and DescreteEventSimulation are the two layers as described previously. The different layers are the single modules of the software. The interface module should ensure the communication and data exchange between the two layers. It is a defined interface between the two layers to make the prototype adaptable for future changes -- if needed.

The RunSimulation module is the overall management module for the software. It gives the trigger for specific functions to the different modules and classes of the simulation. OccupancyMapParameter is the centralized module to hold all input parameters for the simulation. It only holds the parameters. They will not change during runtime.

The discrete event module is divided into sub modules for running the simulation. Each of this sub models is an isolated discrete event simulation. The modules are connected with listeners. This means that a specific event in one module will cause an event in the next module. With these listeners, the different modules are able to communicate with each other.
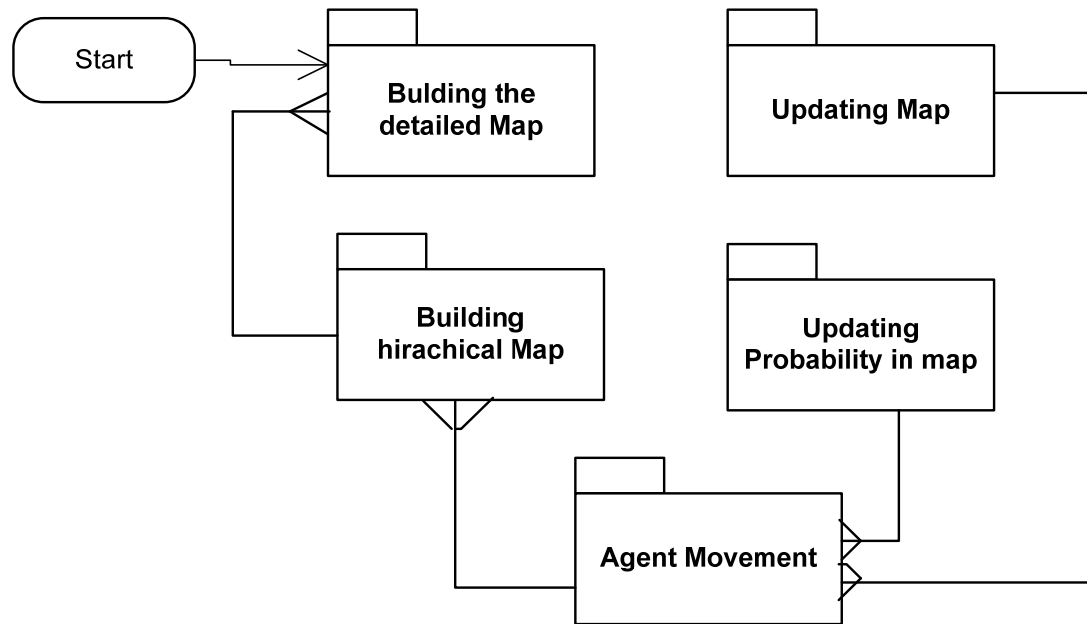

Figure 21.   Sub Modules of the Module Secrete Event Simulation

Figure 21 displays the sub modules of the discrete event simulation. After starting the software simulation, the first module will build the detailed occupancy map for the simulation. If this basic map is finished, the next module will build the hierarchical graph. Both modules will only be used at the beginning of the simulation. They should deliver the hierarchical occupancy map for the later function of the simulation.

The module Agent Movement is responsible for the algorithm of the searcher agent and mover agent. The Simkit-Library has a variety of different kind of movers which will be used in the prototype.

The simulation of the hiding agent and searching agent will influence the probability of each node in the occupancy map. Therefore, the module Update Probability in map must ensure updating of probabilities of the nodes during the runtime of the simulation.

The dynamic behavior of the hierarchical map will be implemented in the module Updating Map. This module listens to the mover module and implements the dynamic behavior -- especially the PopUp and PopDown algorithm in the simulation
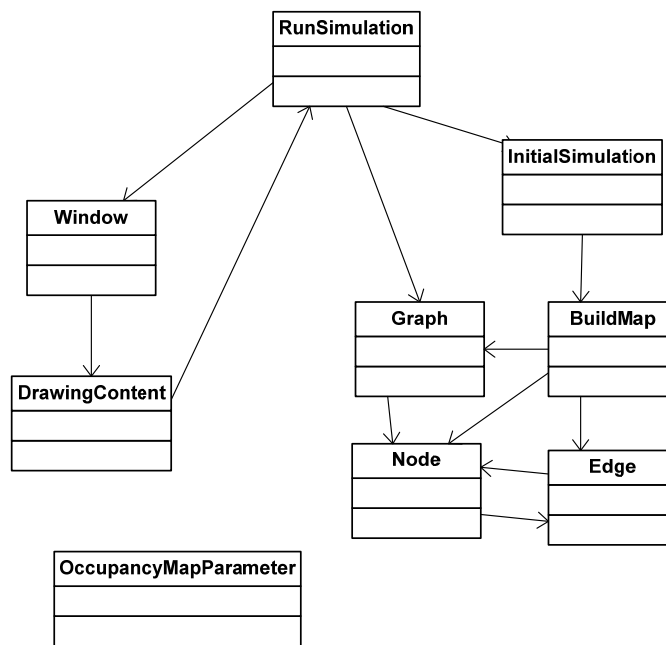

Figure 22.   Class Diagram of the Prototype

Figure 22 shows the different relations of the different classes. The class which is the controller of all classes is the class RunSimulation. This class calls all other classes -- if necessary -- and receives the necessary data from other classes. The class OccupancyMapParameters is the holder of all parameters for

the simulation. It will deliver on all other classes on demand. The parameters are defined as constants and will not be changed during runtime.

Window and DrawContent are the classes of the Graphical User Interface. The class window is the static frame of the prototype. Class DrawContent is responsible for refreshing the content of the display.

2.   Event Graph

The Initial Simulation Class is the trigger to lead all parameters for the Discrete Event Simulation and it starts the simulation by building the basic map. This map consist of the graph with nodes and edges which are all own classes. In Figure 23, the Event Graph for building the map is displayed.
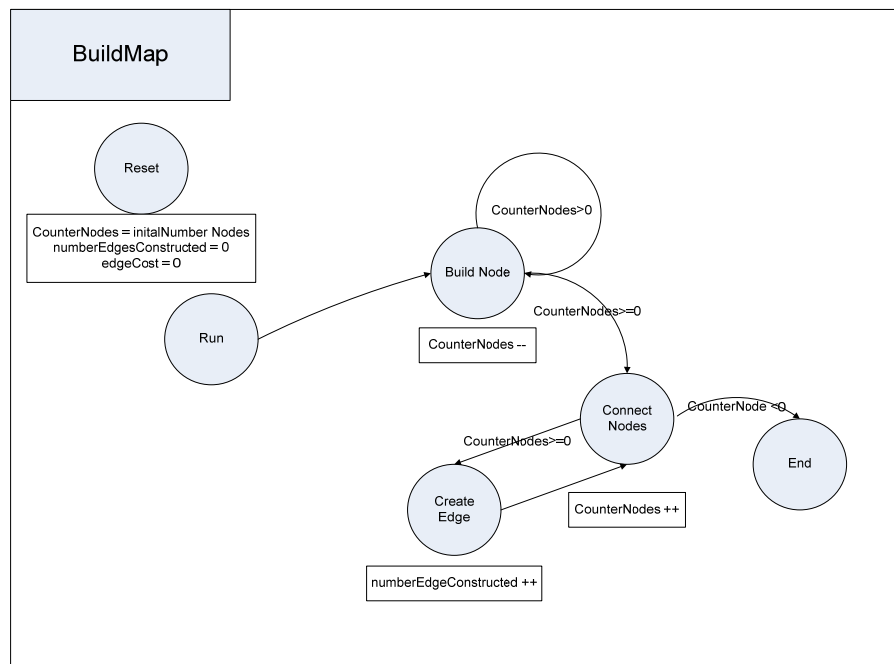

Figure 23.   Event Graph Building Hierarchy Map

The Event Graph shows the building of the basic map. After the Run-Trigger, the first event is to build a specific number of nodes in the basic graph.

Second step is to connect the nodes over the edges. If all nodes are connected, the building of the basic map is finished and the end-event triggers the next part of the simulation.

SearchAgentMovement

Reset

Speed = 0.0;
Map = getOccupancyMap

StartAgent
Movement

EndMove

nextMode = random
(currentNode)

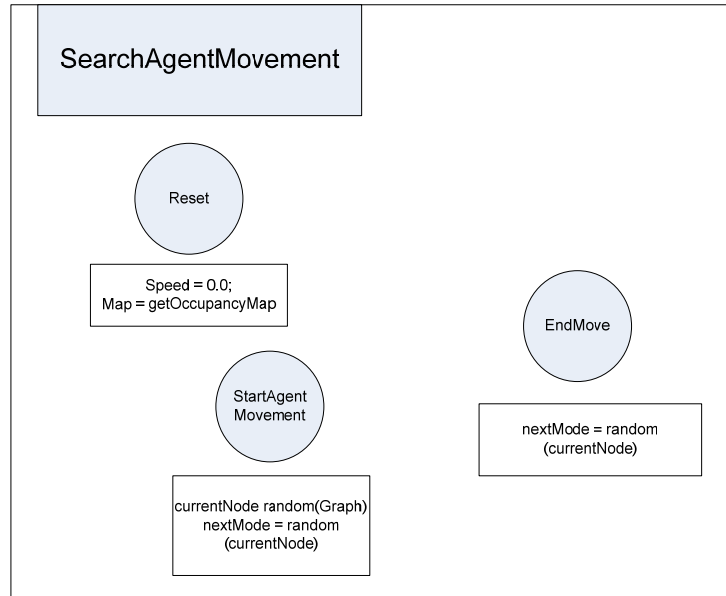currentNode random(Graph)
nextMode = random
(currentNode)

Figure 24.   Event Graph Search Agent Mover

The behavior of the search agent is displayed in Figure 24. With the Simkit-Library, the mover needs only to start a trigger and the target. When the mover starts with moving, the Event StartAgentMovement will be fired. If the agent reaches the target, the Event EndMove will be set on the Event List and the algorithm for searching the next target will work. This next target will be delivered to the new start movement command. There is no connection between the events. This is specific for the class of movers in the Simkit-Package [11]. The events or messages will be sent from the package to the event level itself, when the specific event, here arriving of the target occurs.

An additional element is the Pinger Class. This class is responsible for synchronization or the simulation with the real-time in the discrete event simulation.

C.   IMPLEMENTATION

The simulation will be created by using the Simkit-Package, especially for creation of the Graphic User Interface and the Discrete Event Simulation [10]. Simkit is a library in the programming language JAVA which supports component-based discrete event simulation. In the Simkit-Package, the simulation time does not depend on the real-time in the computer; rather, it has its own time step and internal time in the simulation. When an event occurs, related state variables of the simulation of the occupancy map will change The Graphical User Interface, which shows the behavior of the simulation, is implemented in Java (using the package and methods of the Simkit-Package). The number of nodes of the occupancy map and the area of the simulation could be chosen free. Figure 25 shows the starting point of the simulation with a basic grid with high connection.
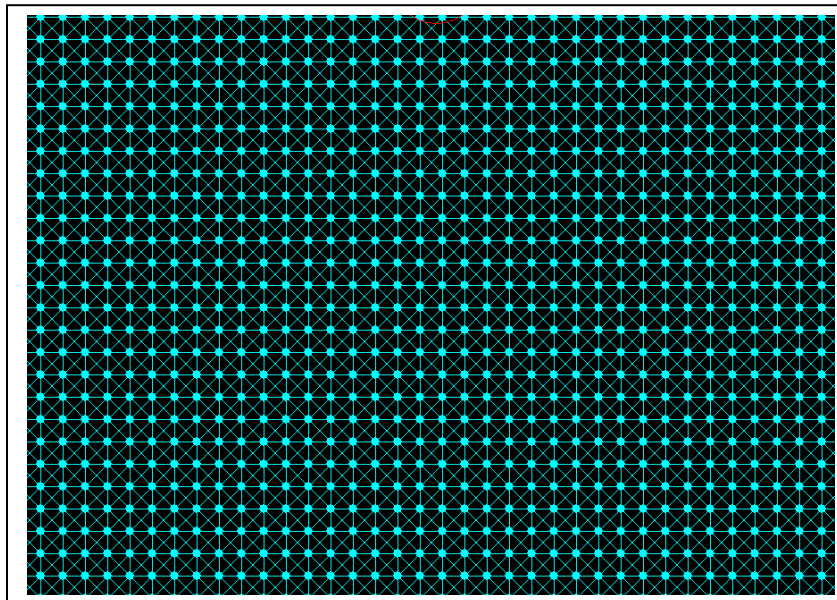

Figure 25.   Basic Graph

The basic graph -- or map -- could have a variable number of nodes with a variable number of edges which connect the nodes. In the building of the prototype, a symmetric graph was used where the nodes are connected to each neighbor.

The agents are displayed as movers in the simulation. In Simkit, movers are entities which can move over the screen [11]. Before they can move, they need target and speed data. If they reach the target, the movers will deliver a message to the Discrete Event Simulation with a time step. In the simulation, two movers are used to simulate the different agents. One is the searcher, who has his own sensor suite and, the other, is the hider, who hides from the searcher on a node outside the sensor range of the searching agent. The searcher agent will be equipped which a simple sensor with a specific range. The hiding agent will receive no sensor.
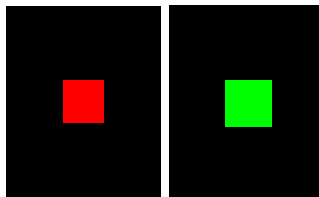

Figure 26.   Movers in Simkit

In the upper picture, the red mover is the search agent and the green is the hiding agent. The hiding agent has no sensor and will only hide. The searcher has a sensor of a specific range which is displayed in Figure 27 as red circle.

Additionally, the red searcher has a white circle. The circle is the marker for the distance from the searcher the hierarchal map will go from the upper hierarchy to the lower hierarchy.
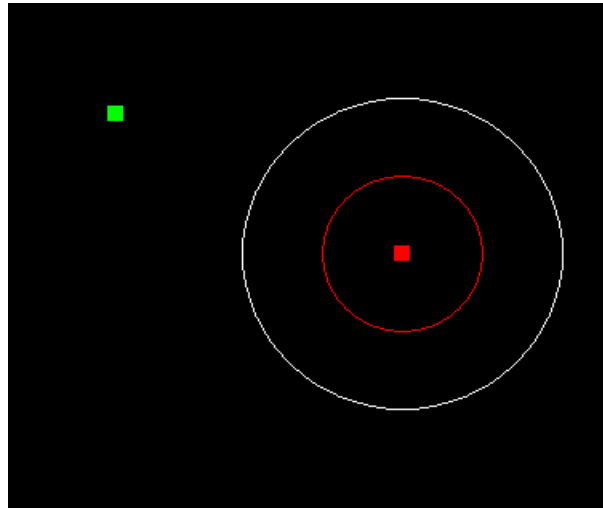
Figure 27.   Search and Hide Agent

This distance, shown in the upper figure as a white circle, is the area the simulation search will work with the detailed map. If a clique of the basic map leaves this area, the single nodes will be mapped together to a node on the next hierarchy. On the other hand, if a node of the upper hierarchy enters this range from the searcher agent, it will be dispersed in the single detailed nodes of the clique.

Between these two areas of the map, there is a transition area which connects the upper hierarchy level to the lower hierarchy level. For the connection itself, temporary edges will be used. Temporary means that these edges will be created and, if the transition area changes, they will be deleted during simulation runtime. On the one end of the edge is a node of the basic node and, on the other end, a node of the hierarchy level.

The cost or parameter of the edge must be calculated from the parameters which originally connects the cliques of the basic graph which underlay the transition area.
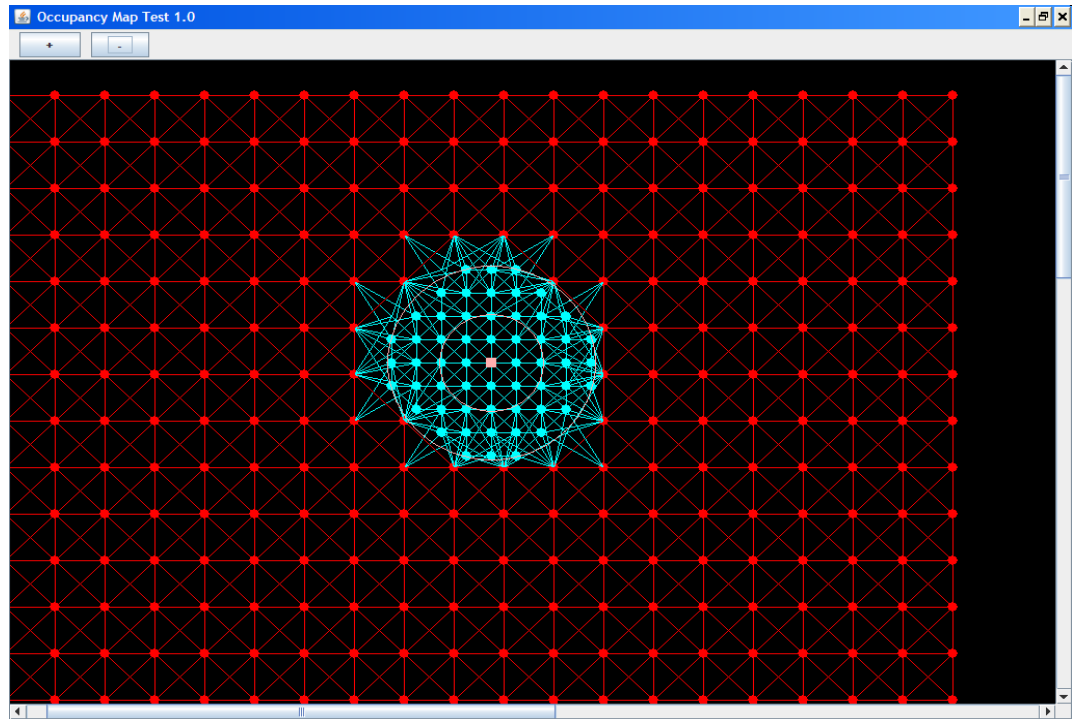
Figure 28.   Search Agent in the Hierarchic Map

Figure 28 shows the first version of the prototype. The red graph is the upper hierarchy which is symmetric. The green is the detailed graph which is only active in the outer circle of the agent.

The search agent is in the middle of the detailed area. This area wanders with the circle over the upper hierarchy. The transition areas are on the outer sides of the circle. The edges are in green and connect the red nodes of the hierarchy map with the green nodes of the detailed map.

If the searcher moves off the detailed map, the current edges, which connect both levels, will be deleted. New edges in the transition phase will be created during simulation runtime. Such deletion and creation, however, will only occur if there is need to bring a hierarchy node to a detailed level. It will also occur to bring a detailed set of nodes in a clique in an upper level.

Thus, this process is a part of the overall PopUp-PopDown process of the nodes in the simulation.
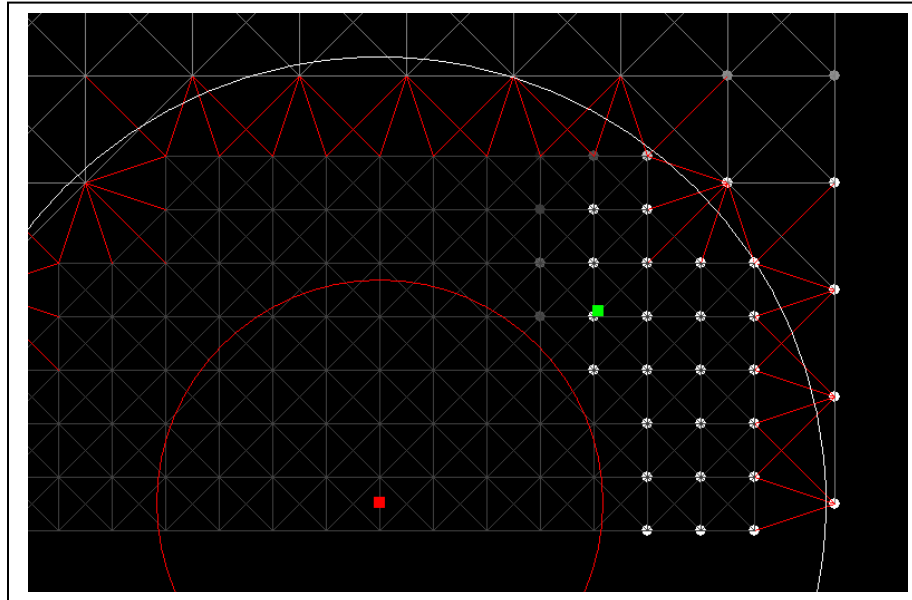

Figure 29.   Distribution of Probability

Figure 29 shows the final version of the prototype. The searcher has the fine granularity in the inner grid. In the outer grid, the graph has a wider granularity. The searcher (red dot) has the inner circle as marker –no matter if the hider is visible or not. This red circle is important for the distribution of the probability and the move-cull process.

All nodes, which are in the red circle, are visible to the search agent in the simulation. The nodes of the basic map and the nodes of the hierarchy map, which are outside of this circle, are not visible to the searching agent.

The green square is the hiding agent. It is also only visible to the searching agent if the hiding agent is in the red circle.

The white circle is the border for the basic map. Outside the hierarchy map, it switches to the higher level. Inside, the hierarchy map switches with the

lower level. The red edges on the border for both levels connect these hierarchies together. These are the temporary nodes as described previously.

The nodes for the hierarchy graph have different colors. The color will represent the probability for the place of the hiding agent via the sight of the searching agent. White color signals a high probability the hiding agent will hide at this place. The less the probability of hiding at a specific node, the darker the node. If the probability is zero, the node will be black and not visible on the black background.

In Figure 29, all nodes in the red circle which are visible are black. This means that the probability for these nodes is zero. Because of the visibly of the nodes, the cull-process put them to zero. In the right upper area, which is the place of the hiding agent, the probability is high (white color). It happened because the hiding agent left the visible area of the searching agent in this direction.

For running the prototype with the simulation, special parameters are needed. The initial type of parameters that the prototype will run are the following.

Parameter	Value
Area Dimension	1000 x 1000
Number of Nodes	1200
Nodes in Clique	4
Cost of Edge	5
Sensor Range Search Agent	100
PopUp/PopDown Range	200

Table 2   Simulation Parameter

This will ensure there are well-established and equal conditions for the initial test runs. The area's dimension is not too huge or too small to the sensor range and the PopUp/PopDown Range. If the area is too small relative to the sensor range, the behavior of the probability distribution is not in a huge simulation. If the area is too huge, much computational costs are expended during the initial tests.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ANALYSES OF THE DYNAMICAL HIERARCHICAL OCCUPANCY MAP

In this chapter, the initial tests and analyses of the prototype will be described. The first analysis is the visual description of a typical situation. This occurs during the run of the agent-based simulation with the searcher agent and hiding agent.

In the second part of the analyses, data collection is performed and, then, probability is distributed. The cull-procedure and its influence on the probability distribution will be considered. With diagrams, some difficulties with the probability distribution could be discovered.

A. VISUAL

The function of the prototype with different scenarios is tested. In these scenarios, it will be discovered, from the algorithm, if the simulation works as specified and expected. The primary focus during the test will be on the probability distribution over time. In the prototype, the probability is colored with grayscale. A white dot on the node means a high probability; a greyer or black dot means a low probability. Black dots will not be visible on the simulation. This means that the probability is zero or very small.

1. Visible Hiding Agent

The first scenario displays a typical configuration in an agent-based simulation with an occupancy map. The searching agent searches for the hiding agent who is visible to the searcher.
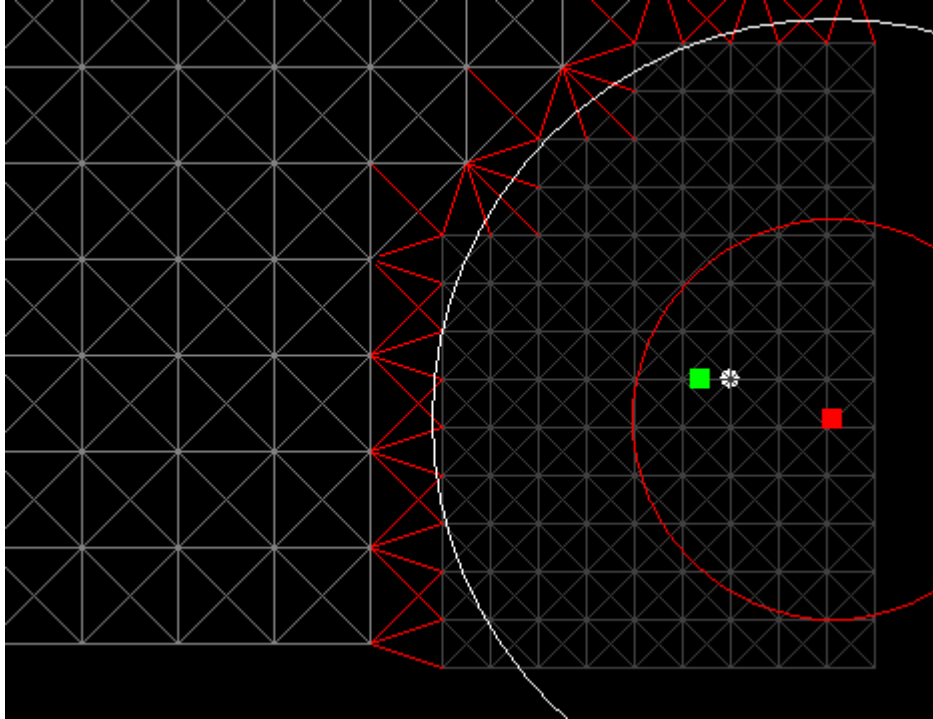
Figure 30.   Hiding Agent Visible

The green hiding agent is in the range of the red searching agent. The hider moves randomly on the grid of the basic map. As specified in the prototype, the searching agent should track the hiding agent. For the searching agent, it is the simplest scenario. The algorithm for this use is only search the nearest node to the hiding agent, set this node to the probability of one, and set all other nodes to the probability of zero.

As displayed in the upper hierarchy, the algorithm works. The white dot, which shows a very high probability, is near the green hiding agent. If the hiding agent moves to this other node during the test, it jumped from one node to the other node. There is no additional coloring of probabilities on the whole graph. This means the probability on other nodes on the graph is very small or equal to zero.

The searching agent moved during the detection of the hiding agent. As expected, this did not change the coloring of the nodes. The estimation of the location of the hiding agent does not depend on the movement of the searching agent. This is because, if the hiding agent is visible, the location is clear. The PopUp/PopDown function during the run time worked. If some nodes left the circle for the inner grid, a new hierarchy node was displayed and the nodes of the basic grid disappeared. Also, new temporary edges between both levels of nodes appeared. As expected from the algorithm, this dynamic behavior of the hierarchal occupancy map had no influence on the single node with high probability.

2. Leaving the Visible Area

If the hiding agent leaves the visible area of the searching agent, the second scenario happens. The hiding agent moves away form the searching agent and, when arriving at a specific range which is out of the sensor range of the searching agent, the distribution of the estimation of location will start. This is, then, the ordinary move-cull process as described in Chapter III.

First step in this process is that the probability of each node, which is at this point only at one node, will be distributed over the edges to the neighboring nodes. The amount of the probability of each node, which should be distributed is a specific parameter, and also depends how often the distribution process will be called.

The second step is the cull-process. It deletes the probability of all nodes which are visible to the searching agent.
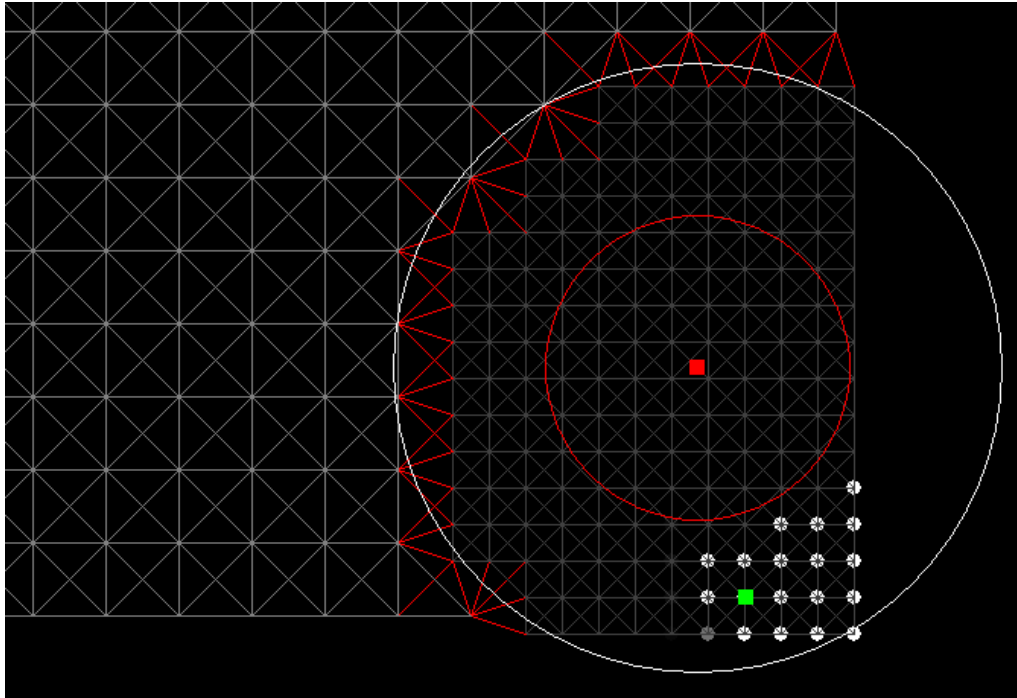
Figure 31.   Probability Distribution Short after Leaving Visible Area

Figure 31 gives an example of this behavior. The green hiding agent just left the visible area of the red searching agent. The distribution of probability should start then.

As we see around the green hiding agent, there are some white dots. This means from the leaving point of the hiding agent from the visible range of the searching agent, there is high probability that the hiding agent is at one of the surrounding nodes. The estimation of the searching agent is displayed as white and grey dots. The white dots are near the hiding agent. Thus, because the hiding agent in real is, the searching agent estimates it.

The result is as expected. After leaving the visible range, there should not be too many nodes with high probability. All other nodes are dark grey or black. After these nodes leave, their distribution of probability seems correct. At his time, there should not be too many nodes with a high or medium probability.

Such nodes would indicate that the distribution of the probability is too fast and not comparable to the moving-speed of the hiding agent.

3. Outside the Close Area

The third scenario is when the hiding agent was not visible to the searching agent for some time. As expected, there is a wide distribution of the probabilities over the occupancy map. If the time is not too long, only a specific part of the occupancy map should have a higher distribution of the probability. The close area of the searching agent should have no probability. This is because the cull-process should set all visible nodes to zero.

The underlying process is the same as in the previous chapter: the distribution process spreads the distribution over the map and, after that, comes the cull-process.

It is important that in the transition region, which connects both levels of the occupancy map, is no barrier for distributing the probability. The distribution should also happen on the map with high granularity.
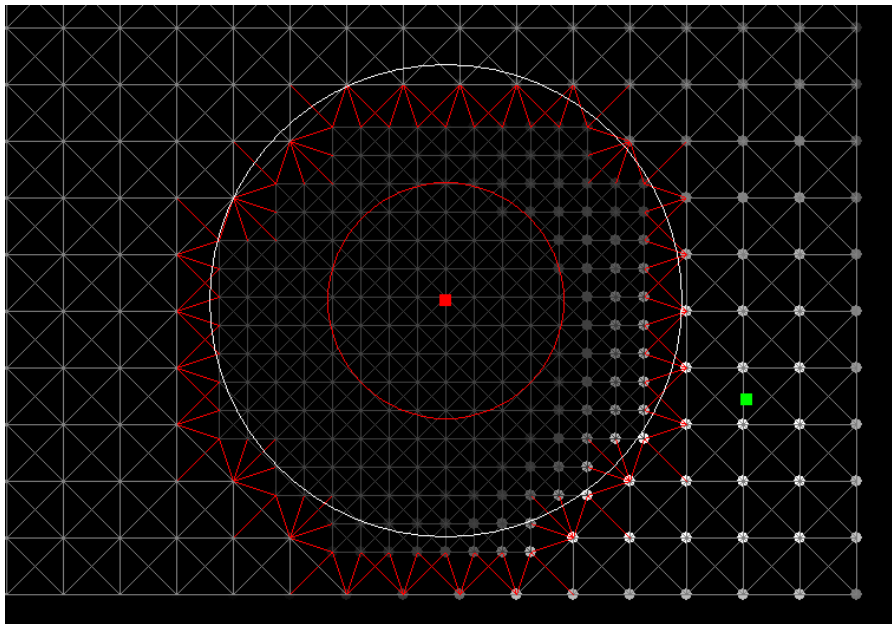

Figure 32.   Probability Distribution Hider outside fine Granularity

Figure 32 gives an example for such a scenario. The hiding agent left the visible area some time ago and the searching agent has not found the hiding agent. The probability is widely distributed on the right area of the occupancy map. The hiding agent moves in the area of the upper level of the hierarchical occupancy map and. The probability distribution reached this area as it followed the distribution works over the transition area with only temporary edges.

Notable is the fact that about the transition areas the nodes of the upper hierarchy have a higher probability as to the node in the lower hierarchy which whom they are connected. The reason is these nodes were built with the dynamic behavior of the map. This was recent and it summed the probability of the four nodes of the clique in the lower hierarchy.

4. Behavior after Long Time

The last scenario occurs after additional time when the searching agent not found the hiding agent. Expected is a wide spread of the probability over the whole map with similar values of p. Only the areas around the searching node should have less probability. This is because these nodes are visible -- or were visible for a short time in the past.
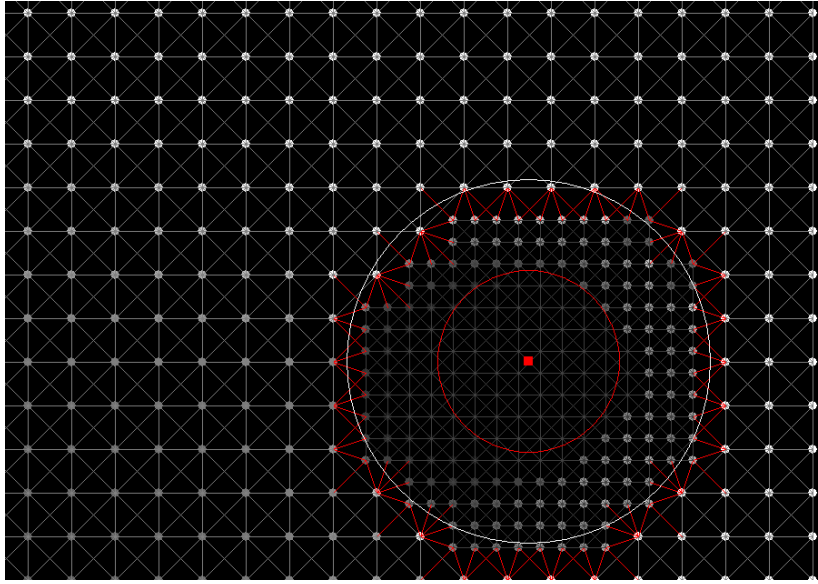
Figure 33.   Probability Distribution after Long Time

Figure 33 shows such a scenario. After some time, the hiding agent is in a completely different area of the occupancy map and not visible in this figure. Many dots are white which is the indicator for a widespread probability over the complete hierarchical occupancy map.

In the inner circle, the nodes are dark. This is because they are visible and, therefore, culled to the zero value of the probability. The right side of the basic grid has nodes with higher probabilities than the left side. The reason is the searcher agent moved from the left to the right side. Following the nodes on his left side was recently visible and the probability of these nodes was set to zero.

B.   DISTRIBUTION OF PROBABILITY

In the previous chapter, some typical scenarios for the hierarchical occupancy map were analyzed. After looking at the visual distribution of probability, this chapter will measure Psum. As in previous chapters, nodded Psum is the sum of all P values of the nodes and necessary to calculate the probability that the hiding agent is at a specific node.

$$p(node) = \frac{p(v)}{p_{SUM}}$$

With the permanent culling of all visible nodes, the Psum will decrease over time. Only if the hiding agent is visible, the nearest node will set to the value of one. This means a refresh of Psum. In a long search, in which the hiding agent is not long visible for the searching agent, Psum has the tendency to become small. Following also the values for the distribution of the probability has the tendency to become small. How it appears over time in shown in Table 3.

Time Step	Psum
1	1.0
10	0.70
20	0.44
30	0.31
40	0.29
50	0.24

Table 3   Psum Over Time

One time step in the table. Figure 34 is the time the searching agent moves from one node to another node. Figure 34 shows the values from the table and more time steps.
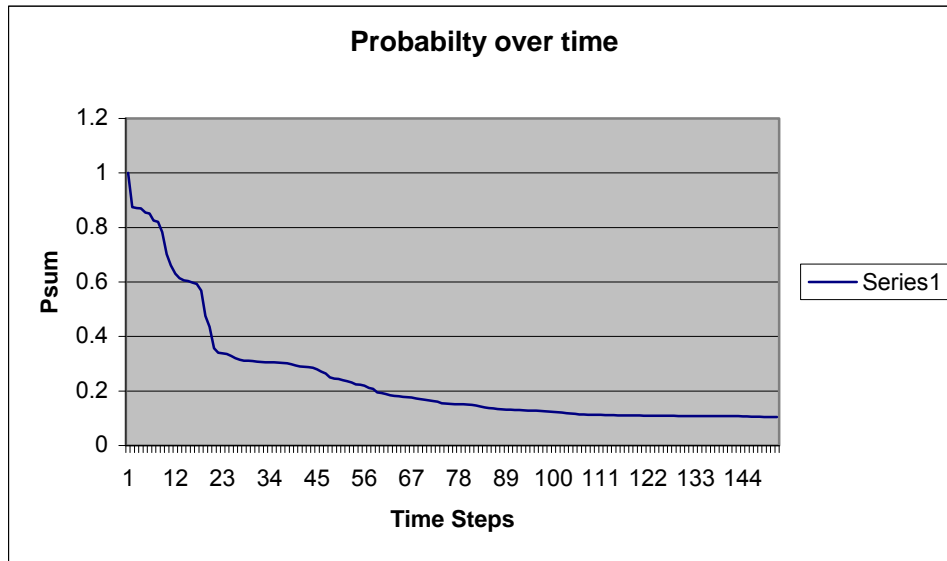
Figure 34.   Psum Over Time

With time, the value of Psum decreases. The decrease, however, is not linear. It appears as a more negative exponential. If Psum is so small, the p values also become smaller. Over time, it is possible that the single p values of the nodes are too small. Thus, there is no good estimation of the hiding agent possible.

However this is not a problem specific to only the hierarchical occupancy map. Of all the maps, this is more problematic because it deals with huge simulation areas. The prototype itself has no algorithm for a periodic refresh of the value of Psum. Such an algorithm could decrease the performance of the prototype itself and therefore was not considered.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION AND FURTHER WORK

A. CONCLUSION

As described in the previous chapters, the prototype is as specified and was used for testing. Some visual analyses were made to proof the algorithm. The result was positive. This means that no error in the prototype occurred. This proofs that the underlying concept of a hierarchical occupancy map could work in every simulation with a huge area.

Some additionally proofs about the probability distribution were made. These test showed some problems with p_{sum} over time. The value of p_{sum} will decrease over time only if the hiding agent is not visible over time. After some time, it is too small to be recognized in the simulation or as data in the computer memory. This is a specific problem, however, of occupancy maps -- not only hierarchy occupancy maps. The solution could be a refresher algorithm after some time without viewing the hiding agent or the acceptance of the problem. The acceptance could be considered as this happens in real-time. If, over time, no one views a hider, no one can estimate its position.

During the work with the prototype, the author of this study became sure that the prototype could be used for a variety of different tests, proofs, and analyses.

B. FUTURE WORK

Given that the concept was proofed as workable, offers a wide variety of future work. There could be different extensions to the underlying concept of a hierarchical occupancy map, the algorithm, or the prototype itself. During this thesis, only some additional tests were made with the prototype and the hierarchy occupancy maps. Therefore, there is a wide and open field for additional tests and improvements of the prototype.

1.   Quantitative Research

There is still a set of quantitative research questions that are unsolved. One of these questions is what is the optimum size required to open a level in the hierarchy maps? With this experimental prototype, someone could prove how significant the reduction of computational cost is. The result could be an improved and optimized algorithm for building hierarchical occupancy maps, especially the dynamic behavior.

Investigating the problem of the Psum would be useful. An implementation of a refresh algorithm could lead to a high computational cost, which consumes the benefits of the decrease of computational cost of the hierarchy occupancy map. On the other hand, to ignore the problem, could lead to a unrealistic behavior of the searching agent, because the agent loose targets with far distance.

2.   Implement in Simulation

With the prototype, the underlying concept of a hierarchical occupancy map was proofed. The next logical step would be to choose a simulation and/or a computer game. The goal would be to improve the tracking behavior of the agents in this simulation. It would be useful to choose an occupancy map that has already been tested and implement it for targeting and tracking. It is important to note that changing the complete tracking and targeting concept of a simulation has an inherent risk. This is significantly reduced when using a simulation with an old occupancy map.

Additionally, this proposed implementation of a simulation of the hierarchical occupancy map should have a large enough area. This is critical to demonstrate the advantages between the same simulation with and without hierarchical occupancy map,

3.   More Hierarchies

A logical additional step to improve the hierarchical occupancy map is additional levels of abstraction. The question is how many levels are useful and, especially, where the optimum distance between such levels is. If there are too many levels in a small area, there is a huge computational cost in the dynamic behavior between the levels. If the levels are too far away, the computational cost would not be reduced significantly enough.

Overall, a hierarchical occupancy map with only two levels will not be significant enough to reduce the computational cost of large simulations. Following up on additional levels is a necessary consideration for future expansions of the prototype.

4.   Hybrid Model

As in the paper of Simulacra described in Chapter I [1], the occupancy map has some disadvantages. The hierarchical occupancy map will not solve these. One is, for example, the magic movement of estimation of probable locations for targets. With a higher granularity in the outer regions of the hierarchical occupancy map, the partiality that such magic movement occurs will increase. There are larger areas without nodes that are only connected with edges.

By combing particle with occupancy map, this problem could be avoided. The particles could wander over the edges from one node to the other. There would be no reason why particles would not also move over the temporary edges.

In such a model, the temporary edges, however, and the PopUp-PopDown algorithm would be the challenge. How should particles which are on the temporary edges during this process behave? The second problem would be the division of the particles if they are located on a node which will be divided in its sub nodes from an upper hierarchy to a lower hierarchy.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1]   C. Darken and B. Anderegg, "Particle filters and simulacra for more realistic opponent tracking," in *Game AI Programming Wisdom 4*, St. Rabin, Ed. Boston, MA: Charles River Media, 2008, pp. 419-42.

[2]   C. Bererton, "State estimation for Game AI using particle filters," in *Proceedings of the AAAI Workshop on Challenges in Game AI*, Technical Report WS-04-04, AAAI Press, 2004.

[3]   D. Borovies, "Particle filter-based tracking in a detection sparse discrete event simulation environment," M.S. thesis, Monterey, CA: Naval Postgraduate School, 2007. [Online]. Available: Naval Postgraduate School – Dudley Know Library, http://bosun.nps.edu/ . [Accessed September 15, 2008].

[4]   B. Ristic, S. Arulampalm, and N. Gordeon, *Beyond the Kalman Filter – Particle Filters for Tracking Application*, Norwood: Artech House Publishers, 2004.

[5]   N. Sturtevant and M. Buro, "Partial Pathfinding Using Map Abstraction and Refinement," in *Proceeding of the Twentieth National Conference on Artificial Intelligence*, 2005, pp. 1392-1397.

[6]   National Research Council, Committee on Modeling and Simulation for Defense Transformation, *Defense Modeling, Simulation, and Analysis: Meeting the Challenge*, Washington D.C.: The National Academies Press, 2006. [Online]. Available: The National Academies Press, http://www.nap.edu/catalog.php?record_id=11726. [Accessed September 15, 2008].

[7]   T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm,* 2nd edition. Cambridge: The MIT Press, 2001.

[8]   Averill M. Law, *Simulation Modeling and Analyses*, 4th edition. New York, NY: McGraw-Hill, 2007.

[9]   Eric Braude*, Software Design – From Programming to Architecture*, Hoboken, NJ: John Wiley and Sons, 2004.

[10]   Arnold Buss, "Component-Based Simulation Modeling with Simkit," in *Proceeding of the 2002 Winter Simulation Conference*, 2002, pp. 243 – 249.

[11]   Arnold Buss and P. Sanchez, "Simple Movement and Detection in Discrete Event Simulation," in *Proceeding of the 2005 Winter Simulation Conference*, 2005, pp. 992 - 1000

INITIAL DISTRIBUTION LIST

1.   Defense Technical Information Center
       Ft. Belvoir, Virginia

2.   Dudley Knox Library
       Naval Postgraduate School
       Monterey, California