NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ADVANCED TERRAIN REASONING AND AUTOMATED MANEUVER PLANNING

by

Peter Severson

June 2019

Thesis Advisor: Imre L. Balogh
Second Reader: David Reeves (MOVES)

Research for this thesis was performed at the MOVES Institute.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ADVANCED TERRAIN REASONING AND AUTOMATED MANEUVER PLANNING	**5. FUNDING NUMBERS** RVLXU
6. AUTHOR(S) Peter Severson	
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) OAD, Quantico, VA 22134; CD&I, Quantico, VA 22134; OAD, Quantico, VA	**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** A

13. ABSTRACT (maximum 200 words)

Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is the primary analytical combat simulation tool used by the Marine Corp's Operations Analysis Directorate (OAD) and The Research Analysis Center (TRAC) under the new U.S. Army Futures Command (AFC). While the simulation has considerable capabilities and has been used for diverse studies, it has two major shortcomings that can be addressed by innovative model development techniques. The first weaknesses of COMBATXXI at the scenario development level is maneuver planning. An entity's maneuver is completely scripted by scenario developers. This is a time-consuming process that varies in realistic accuracy based on the developers' expertise and experience. Advanced terrain reasoning in COMBATXXI could enhance dynamic decision making by simulated entities, improve analysis of combined arms operations, and shorten the scenario development time. The second weakness is prototyping and testing new behaviors in COMBATXXI. This is a difficult process that can be improved by using a simplified surrogate environment. This thesis took one step toward addressing both shortcomings by developing a prototype maneuver planner in a surrogate environment (WOMBATXXI/Unity3d Game Engine) using an advanced terrain reasoning approach, testing the planner output through a simple face validity process of examining the realism and simplicity of the plans, and demonstrating the planner output over five distinct scenarios.

14. SUBJECT TERMS terrain reasoning, maneuver planning, tactical position selection, artificial intelligence, COMBATXXI, modeling, simulation	**15. NUMBER OF PAGES** 125
	16. PRICE CODE

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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ADVANCED TERRAIN REASONING AND AUTOMATED MANEUVER PLANNING

Peter Severson
Captain, United States Marine Corps
BS, U.S. Naval Academy, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION

from the

NAVAL POSTGRADUATE SCHOOL
June 2019

Approved by: Imre L. Balogh
Advisor

David Reeves
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is the primary analytical combat simulation tool used by the Marine Corp's Operations Analysis Directorate (OAD) and The Research Analysis Center (TRAC) under the new U.S. Army Futures Command (AFC). While the simulation has considerable capabilities and has been used for diverse studies, it has two major shortcomings that can be addressed by innovative model development techniques. The first weaknesses of COMBATXXI at the scenario development level is maneuver planning. An entity's maneuver is completely scripted by scenario developers. This is a time-consuming process that varies in realistic accuracy based on the developers' expertise and experience. Advanced terrain reasoning in COMBATXXI could enhance dynamic decision making by simulated entities, improve analysis of combined arms operations, and shorten the scenario development time. The second weakness is prototyping and testing new behaviors in COMBATXXI. This is a difficult process that can be improved by using a simplified surrogate environment. This thesis took one step toward addressing both shortcomings by developing a prototype maneuver planner in a surrogate environment (WOMBATXXI/Unity3d Game Engine) using an advanced terrain reasoning approach, testing the planner output through a simple face validity process of examining the realism and simplicity of the plans, and demonstrating the planner output over five distinct scenarios.

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

LIST OF ACRONYMS AND ABBREVIATIONS

AA	avenue of approach
ABPS	automated battle planning system
AFC	Army Futures Command
AI	Artificial Intelligence
BSL	Behavior Scripting Language
CD&I	Combat Development and Integration
COMBATXXI	Combined Arms Analysis Tool for the 21st Century
HTN	Hierarchical Task Network
MCCDC	Marine Corps Combat Development Center
MCDP	Marine Corps Doctrinal Publication
MCRP	Marine Corps Reference Publication
ME	main effort
METT-T	mission, enemy, terrain, troops available - time
NPS	Naval Postgraduate School
OAD	Operations Analysis Directorate
OAKOC	obstacles, avenues of approach, key terrain, observation and fields of fire, cover and concealment
PCC	pre-combat check
PCI	pre-combat inspection
SE	supporting effort
SE1	supporting effort #1
SE2	supporting effort #2
TCM	Tactical Control Measure
TPS	Tactical Position Selection
TRAC	The Research Analysis Center
WXXI	WOMBATXXI

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This research would not have been possible without the help of the MOVES faculty and several individuals who poured time and energy into the development of this work. Specifically, I would like to thank Dr. Imre Balogh, Dr. Christian Darken, and David Reeves. Their guidance and experience have been instrumental.

The most important acknowledgment and thanks go to my wife, Dana. Her support and understanding allowed me to focus and dedicate myself to completing this program and this research. I especially appreciate her willingness to let me work at any time regardless of the circumstances while she carried the burden.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

An important use of combat simulations is to support analytical studies used in the decision-making process. Key leaders use analysis gained form combat simulations to aid in acquisition decisions, policy decisions, and many others. As the operational environment evolves and becomes increasingly complex, the reliance on combat simulations will only increase. In order to keep up will the changing operational environment, we must continue to improve the efficiency and realism of combat simulations. This thesis seeks to address two shortcomings of a popular combat model that will improve efficiency and realism.

A. COMBATXXI

The Combined Arms Analysis Tool for the 21st Century, or COMBATXXI (CXXI), is an entity level, high-resolution, closed-form, stochastic, discrete event-driven simulation that focuses primarily on the analysis of ground combat [1]. While its primary focus is ground combat, CXXI does provide support for amphibious operations including some ships and aircraft.

1. Entity Representation

CXXI provides a high level of detail at the entity level with respect to sensors, weapons, communications, decisions, and actions. Each entity is fully customizable with weapons, sensors, communications equipment, and much more. In addition, an entity's perception of its environment based on its sensor capability is affected by the various models that represent physical phenomena within the environment (weather, sound, etc.).

2. Environment/Terrain Representation

The environment within CXXI is represented as a grid of real-world elevation postings, which allows the model to run combat analysis on real-world terrain data. Entity movement is accomplished through assigning scripted routes to individual entities or unit leaders. The scripted routes are composed of fixed waypoints that were selected and assigned in a specific order by a scenario developer. While a pathfinding capability is

available that would significantly reduce the scenario development time, it is rarely used by scenario developers, as their ability to shape the scenario will be reduced [2].

3. Simulation Engine

CXXI uses a discrete event simulation to model time. A discrete event system models the simulation in states that change based on the occurrence of an event. Those events then trigger linked events, which again change the state of the system. This produces a generally continuous model of time. While the simulation engine is a major component of any simulation, it is not the focus of this research. The basic concept explained above should be adequate to understand the plans system explained below and its relationship to the scenario development process.

4. Plans System

COMBATXXI's plans system is structured similar to a traditional military structure were orders or instructions are passed down from higher-level units to subordinate units. Formal plans in CXXI are executed as coordinated maneuver orders where orders can be issued to any unit within a user defined command hierarchy. Those orders trigger subsequent orders to that specific unit and potentially subordinate units creating a chain of orders that result in unit actions defined as behaviors. Behaviors in COMBATXXI are created by the developer using either Behavior Scripting Language (BSL) or Python code. The Behavior Studio package, which allows developers to create behaviors, uses a tree structure that can be embedded with conditional check and Python code to guide execution when the behavior tree is invoked by an order or event.

Altogether, the system provides a robust representation of entities and their environment from which scenarios can be developed and customized, and analysis can be conducted. However, the complexity of each module, the heavy burden on developers to manually script the majority of the systems functionality, and the long scenario development times makes CXXI ideal for improvements from automated scenario development processes. This thesis will provide one solution to terrain analysis processes that can be automated to support maneuver planning and scenario development. The

methodologies developed here can be used to assist in scenario construction or to enable dynamic planning during scenario execution.

B. ORGANIZATIONAL USE

COMBATXXI is the primary analytical tool used by the Marine Corp's Operations Analysis Directorate (OAD) under Marine Corps Combat Development Command, Combat Development and Integration (MCCDC/CD&I) and The Research Analysis Center (TRAC) under the new U.S. Army Futures Command (AFC). CXXI is used by these centers as a comparative analysis tool to inform decision makers on a variety of topic including acquisitions, technology development, and force employment. Understanding of a particular problem set is gained through numerous repetitions of a scenario and follow-on statistical analysis.

C. SCENARIO DEVELOPMENT ISSUES

While COMBATXXI provides a robust environment to conduct analysis, it also has some weaknesses that limit its effectiveness. This thesis will focus on two weaknesses of the simulation. The first weakness of COMBATXXI is maneuver planning. In this context, we use maneuver planning to mean position selection and route planning. An entity's maneuver is completely scripted by scenario developers. This process varies in realistic accuracy based on the developer's expertise and experience. Additionally, maneuver planning is a time-consuming process that limits the number of a simulation runs that can be accomplished in a given time.

Tactical pathfinding, terrain reasoning, and the like have been around for some time and used extensively within the artificial intelligence (AI) and gaming community. However, it has had negligible use in COMBATXXI and the Department of Defense modelling and simulation community. Institutional resistance to automated maneuver planning has been present based on the statistical variability associated with automated behaviors. Added variability requires an increase in the number of simulation runs to show statistical significance between alternative solutions. Currently, developers and analysts at OAD and TRAC do not have the available time to test and implement an automated maneuver-planning tool and accomplish the necessary runs to account for the variability in

automated planning. Automated maneuver planning will reduce the up-front scenario development time and provide analysts the additional time needed to do simulation runs.

The second weaknesses of COMBATXXI is the difficulty of prototyping and testing in the complex development environment. While COMBATXXI is a detailed simulation that models many combat related processes including target acquisition, communication, and engagement, testing prototypes in COMBATXXI is no easy matter. COMBATXXI relies heavily on loggers to output data, which can be difficult for users to decipher. This issue was explored further by Miller who developed a solution for prototyping hierarchical task networks (HTNs) in Unity3D for development and testing prior to implementation in COMBATXXI [3].

D. UNITY3D AS A SURROGATE ENVIRONMENT

Unity3d is one of the most popular commercial game development environments that provides users with a set framework and broad array of features to build games efficiently [4]. Unity is specifically popular because it has an interactive and user-friendly development environment encompassed in a single application window, as well as a large community of users that provide a wide variety of support in the form of forums and formal documentation.

E. WOMBATXXI DEFINED

WOMBATXXI (WXXI) is a combat simulation built in the Unity3d game engine that resembles COMBATXXI in its framework. WOMBATXXI was developed at The Naval Postgraduate School (NPS) by LtCol Byron Harder as part of his dissertation *automated battle planning for combat models with maneuver and fire support* [2]. WOMBATXXI provides a good platform to prototype behaviors for COMBATXXI prior to implementation without having to deal with the complexity of COMBATXXI. This thesis will leverage WOMBATXXI to explore maneuver planning in a framework similar to COMBATXXI. Unity3D, and more specifically WXXI were chosen as the surrogate environment for this thesis based on resident expertise, formal documentation, no cost burden, and a framework representative of COMBATXXI. See Chapter IV for further exploration of WXXI.

F. BENEFITS OF THIS THESIS

The overall goal within COMBATXXI is that entities are able to dynamically make decisions within the simulation. This will reduce the scenario development time and improve analysis of combined arms military operations. The development of an automated maneuver planner is one step toward this goal. The reduction in scenario development time and improved analysis will benefit Marine Corps and Army developers at both OAD and TRAC.

The ability to prototype behaviors in Unity3d and transfer those behaviors into COMBATXXI will allow developers to test new behaviors in an efficient manner without interrupting project timelines and adding unnecessary burden on scenario developers. This will allow for more efficient development and implementation of new behaviors in COMBATXXI.

THIS PAGE INTENTIONALLY LEFT BLANK

II. REVIEW OF USMC PLANNING

A. DOCTRINAL MANEUVER PLANNING

To begin a discussion about maneuver planning, we must first narrow down our scope to the specific level of war we are operating in and the type of operation we are planning.

1. Levels of War

As defined in Marine Corps Doctrinal Publication (MCDP) 1–0 Marine Corps Operations, there are three levels of war: strategic, operational, and tactical, which align the national strategic objectives to tactical level actions [5, 1–6]. The strategic level involves defining national level goals and assigning resources to meet those goals. The operational level links the tactical employment of forces to national objectives through campaigning [5, 1–7]. The tactical level focuses on planning and executing battles and engagements to achieve military objectives assigned to tactical units [5, 1–7]. More simply, strategic level seeks to win wars, operational level seeks to win campaigns, and the tactical level seeks to win battles.

2. Type of Operation

Each level of war has operations that when executed, help meet the objectives of that level of war. The type of operation drives the planning considerations and execution. While some types of operations are found in all levels of war (ex: offensive and defensive operations), others may be more prevalent in a specific level of war (ex: expeditionary operations are generally operational or strategic level). In addition, each type of operation may have more clearly defined subordinate operations. For example, offensive operations include movement to contact, attack, exploitation, and pursuit (See Figure 1 for visualization).

For the purposes of this thesis, I will focus on the tactical level of war and offensive operations, specifically the deliberate attack. A deliberate attack is characterized by pre-planned coordinated employment of firepower and maneuver to close with and destroy the

enemy [5] (i.e., it requires information about the enemy location, disposition, and strength).

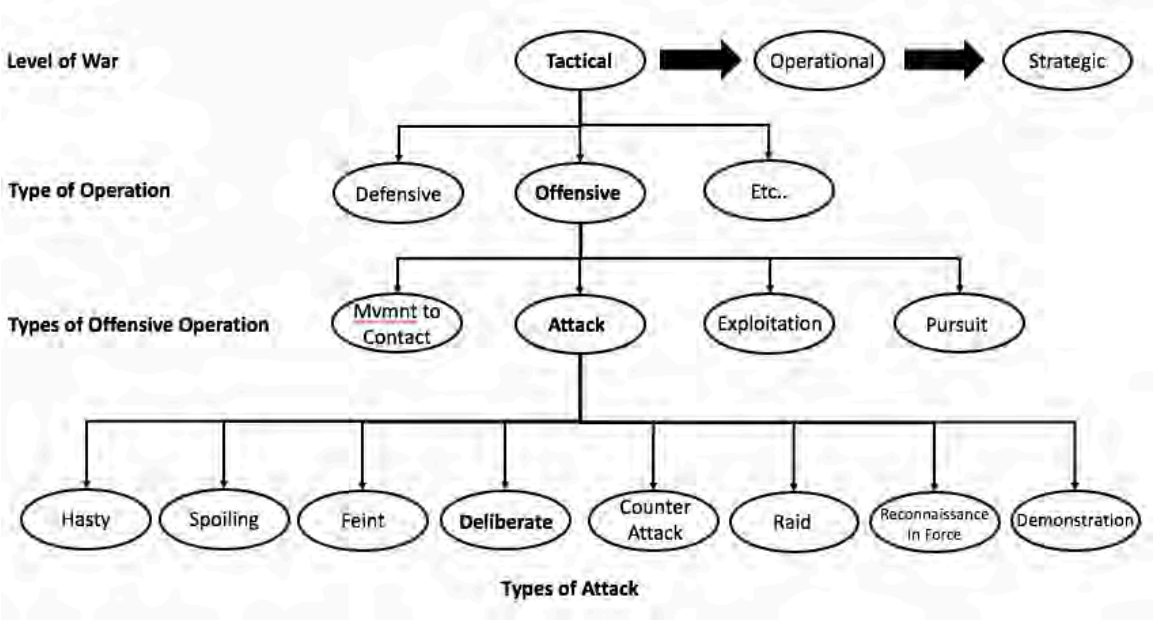


Figure 1. Breakdown of the Level of Wars and Types of Operation

3. Effective Maneuver Planning

During any planning effort, three types of situational factors drive the planning effort [6, 3–5]. The first type is external factors that stem from the environment. Second are internal factors that stem from the availability and expertise of the planners themselves [6]. Lastly are task-related factors relating to the specific circumstances of the operation being planned. While the factors can be examined separately, they are often interrelated and potentially in conflict with each other. Every situation is different and requires different planning considerations. However, Marine Corps warfighting doctrine is based on the view that war is a "time-competitive, interactive clash characterized by high levels of friction, uncertainty, disorder, and unpredictability" [6, 3–7]. Effective maneuver planning under these conditions favor "simple, modular, flexible, and timely plans" [6, 3–7].

4. Estimate of the Situation

The foundation for building an effective maneuver plan is the estimate of the situation based on the mission, enemy, terrain and weather, troops and support available, and time available (METT-T). The estimate of the situation is a brain storming process where all available information is considered and analyzed to frame the situation within which the maneuver plan is built. The estimate of the situation drives the scheme of maneuver (maneuver plan) and the overall concept of operations. A poor estimate of the situation will likely result in an ineffective plan that produces a poor operational outcome.

B. TERRAIN ANALYSIS IN OFFENSIVE OPERATIONS

In order to understand the relationship between terrain and tactical control measures, we must first understand how terrain affects offensive operations, which tactical control measures are important in offensive operations, and how those control measures are defined within Marine Corps Doctrine.

1. Terrain as a Part of Tactics

To define how terrain effects operations, it is important to understand where terrain fits in the key concepts that define Marine Corps Tactics. The key concepts that define tactics can be broken down into six main categories: achieving a decision, gaining advantage, being faster, adapting, cooperating, and exploiting success [7, p.11]. The successful execution of any operation relies on the creative application of all six of these concepts, and potentially more situational dependent concepts.

Terrain is one of the fundamental factors in gaining the advantage. Terrain effects maneuver and tactical formations. In gaining the advantage, a unit attempts to employ tactics that make terrain an advantage for them and a disadvantage for the enemy [7, p.43]. This includes an advantage in maneuver, visibility, and fires.

It is important to note that these concepts are not isolated ideas, but the successful application of these concepts as a whole is what leads to the desired operational outcome. Because these concepts are combined to reach a desired end state, and simply gaining an advantage does not predict success, the proper use of terrain is indispensable to our overall

goal of winning in combat (i.e., the improper use of terrain can adversely affect a unit's ability to achieve success in other key concepts).

2. Terrain Effects on Operations

Marine Corps Reference Publication (MCRP) 2–10B.1 Intelligence Preparation of the Battlefield/Battlespace defines terrain analysis as, "the evaluation of geographic information on the natural and manmade features of the terrain" [8, 4–4]. The objective of terrain analysis is to predict the impact of terrain on military operations, both friendly and enemy [8]. In order to determine the effect on friendly and enemy operations, the military aspects of terrain must be considered. Terrain in itself, without considering its military aspects, can only be viewed as a neutral factor that neither favors nor hinders either side. Military aspects of terrain are analyzed using the OAKOC model [8, 4–4]. OAKOC is a piecewise analysis that considers terrain with respect to obstacles, avenues of approach, key terrain, observation and fields of fire, and cover and concealment [8].

a. Obstacles

Natural terrain or manmade obstructions can only be considered an obstacle if its effect on a force is to disrupt, block, turn, or fix its [8, 4–4]. Obstacles have different effects on mounted mobility, dismounted mobility, and air mobility. For example, densely forested areas and rubble on the road may have a greater effect on mounted mobility, whereas steep gradients and concertina wire may have a greater effect on dismounted mobility. This thesis will focus on obstacles with respect to their effect on dismounted movement.

b. Avenues of Approach

Avenues of approach (AAs) are routes that a force can use to reach an objective or key terrain. AAs are important because a good maneuver plan depends on the availability of suitable AAs. AAs are evaluated for suitability based on "access to key terrain, degree of canalization and ease of movement, line of communication support, and access to the objective" [8, 4–6]. The simplest example of an AA is single road that leads directly to an objective.

c. Key Terrain

Key terrain is any location or area that if retained, gives an advantage to either force [8, 4–8]. Key terrain is evaluated based on the advantage that either force will gain if that terrain is controlled. The advantage gained is determined by analyzing the other four aspects of the OAKOC model with respect to that piece of key terrain. The historic example of key terrain is high ground that provides observation and/or fields of fire over its surrounding area.

d. Observation and Fields of Fire

Observation is the condition of the terrain that allows a unit to see friendly forces, enemy forces, and important aspects of the area of operations [8, 4–9]. Observation includes the use of sensors and surveillance equipment, but for the purposes of this thesis, I will focus specifically on line of sight. Fields of fire are a specific type of observation that is limited by a unit's weapons capability. For example, a unit may be able to observe its surroundings in a specific direction over a mile away with binoculars or similar sensors, but if its weapons can only effectively engage the enemy out to 800 meters, that unit's field of fire in that direction is considered to be 800 meters. Evaluation of observation and fields of fire help determine weapon dead space and areas where a force is most vulnerable to observation and enemy fires.

e. Cover and Concealment

Cover and concealment are the counters to an enemy's observation and fields of fire. Cover provides physical protection from the effects of enemy weapons and fields of fire. Concealment provides protection from enemy observation. Concealment does not necessarily provide cover. Consider a large rock formation or boulder that can stop enemy rounds and block an enemy's visibility versus a bush that can obscure an enemy's visibility but is not suitable to defend against enemy fire. An ideal piece of terrain provides both cover and concealment during maneuver and in the engagement, up to the point that a person willingly exposes himself.

3. Tactical Control Measures

Tactical Control Measures (TCMs) are what a commander uses to coordinate and synchronize the effects of his combat power. Different types of operations require different control measures. Offensive control measures are usually defined by prominent pieces of terrain that are easily identifiable and offer some cover and concealment. A good maneuver plan will utilize the minimum amount of control measures needed to execute the mission, while maintaining flexibility needed to adjust during the conduct of the operation. This thesis will focus on the five major control measures that every good maneuver plan will have: assembly area, attack position, line of departure, assault position, and objective.

a. Assembly Area

The assembly area is where a unit assembles to prepare for offensive actions. The assembly area is generally located in a friendly or permissive environment, which is safe from the effects of enemy weapons and observation. Common tasks that are conducted in the assembly area are rehearsals, ammunition draw, pre-combat checks (PCCs), and pre-combat inspections (PCIs).

b. Attack Position

The attack position is the last covered and concealed position before crossing the line of departure. Common tasks that are conducted in the attack position are chamber ammunition, deploy into initial tactical formations, final coordination with higher headquarters.

c. Line of Departure

The line of departure is either a real terrain feature such as a ridge, or imaginary line, usually a northing or easting that is used to synchronize the departure of all attacking and supporting units.

d. Assault Position

The assault position is the last covered and concealed position before the objective [9, Appendix R]. The assault position must be an easily recognizable piece of terrain

because it is generally located within the weapons effects range of the enemy. Common tasks conducted in the assault position are deploy into assault formations, call for final supporting fires, and fix bayonets. A unit should spend the minimum amount of time necessary in the assault position.

e. Objective

The objective is a key terrain feature or enemy force that is to be seized or engaged in the assault. In the maneuver plan, our objective location, terrain analysis, and tactical task from higher are what drive the identification of the previous four control measures.

C. THE DELIBERATE ATTACK

A deliberate attack can be characterized as an offensive operation where detailed information about the enemy is known, and maneuver is pre-planned and coordinated. Detailed information about the enemy allows a force to conduct deliberate planning, deploy from an assembly or reserve area, task organize the force specifically for the attack, and rehearse the plan. The purpose of any attack is to defeat the enemy, seize terrain, or both [10, 2–83]. The deliberate attack is one of the most important types of attacks because it provides the attacking force with the greatest opportunity to leverage all available combat power, coordinate pre-planned fires, forward stage resources, mass combat power, and achieve surprise in an attempt to defeat the enemy, seize terrain, or both.

1. Deliberate Attack Planning

While planning considerations for a deliberate attack can be numerous and seemingly unending depending on the amount of situational information available, this thesis will focus on leveraging information about the enemy location and disposition, and relevant information about the terrain to determine key terrain and suitable tactical control measures that will be utilized throughout the phases of the operation.

2. Phases of Offensive Operations

Every offensive operation is broken down into three general phases: preparation phase, conduct phase, and exploitation phase [11, 4–3]. These phases are not always

distinct and clearly separated, but the transition of forces between different control measures can often correspond to the transition between phases. In addition, each phase, as well as the transition between control measures requires different tactical formations that support the objectives of that phase. Figure 2 below depicts the most basic frontal attack scenario and the relationship between the phases, tactical control measures, and formations. Time flows from the bottom of Figure 2 to the top. It is important to note that the selection of relevant tactical control measures and tactical formations differs from situation to situation and may even change during the conduct of an operation.

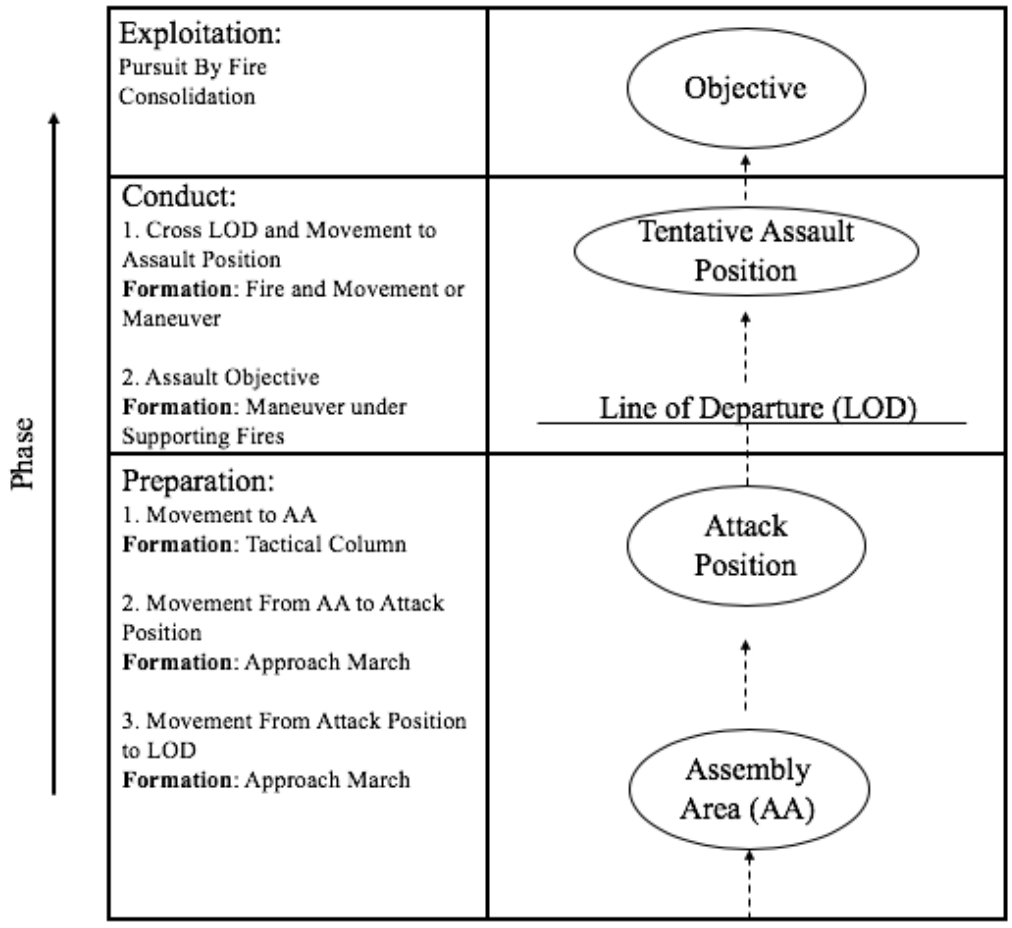


Figure 2. Basic Frontal Attack Scenario Demonstrating the Relationship between Phases, TCMs, and Tactical Formations

3. Basic Terrain Analysis

A basic terrain analysis considers the terrain without looking at its effect on the adversary, and only considers the general impact on friendly forces and the mission. Imagine looking at a map of a specific area of operations and simply identifying prominent characteristics of the terrain and their general effect on any force. At a minimum, a basic terrain analysis should identify the area of operations, significant terrain features (large water features, mountains, urban areas, etc.), and the major environmental factors that constrain friendly operations. For example, a damaged or destroyed bridge restricts a forces ability to cross a large water feature. The basic terrain analysis serves as the foundation for analyzing the effect of terrain on friendly and enemy operations and identifying key terrain and tactical control measures.

4. Terrain Analysis with Respect to the Enemy

In order to select tactical control measures that enable proper coordination of the assault, the terrain must be analyzed with respect to enemy observation, enemy fields of fire, and avenues of approach to the enemy location. This process in the operating forces is commonly referred to as “turning the map around,” which allows a commander to see the battlespace through the eyes of the enemy. An analysis of enemy observation, fields of fire, and avenues of approach allows the attacking unit to identify mobility corridors to the objective area, and pieces of terrain that can provide cover and concealment from enemy observation and fields of fire. These pieces of terrain would then be prioritized with decisive or key terrain at the top, and terrain that provides small benefits at the bottom. A piece of terrain that offers a significant advantage in fires or observation over the enemy may be considered key terrain, whereas, another piece of terrain that simply provides a safe route to a specific area but no significant advantage over the enemy will be lower in priority. The seizure or utilization of decisive or key terrain should be directly implemented in the maneuver plan and potentially used as control measures. Terrain that provides smaller benefits may be used to provide flexibility in the plan, or potentially as control measures. In reality, the terrain analysis, and especially the identification of key terrain should directly drive the development of the maneuver plan.

D. SCOPE OF THIS THESIS

This thesis has two distinct areas of study. The first area of study is the development of a maneuver planner in WOMBATXXI as an addition and complement to the automated fire support planner that is currently implemented. The maneuver planner will focus on identifying the best attack and assault positions, and routes for maneuver elements based on in depth terrain analysis relative to the friendly and enemy force.

The second area of study is an in-depth investigation of the transferability of the maneuver planner prototype from WOMBATXXI to COMBATXXI. Preliminary research in this area has been explored in a previous Naval Postgraduate School thesis: "Hierarchical Task Network Prototyping in Unity3d" by Captain David Miller [3]. His thesis presented a solution for prototyping hierarchical task networks (HTN) by developing HTNs in Unity3d prior to building the HTNs in COMBATXXI. He found that "Developing behaviors in a surrogate environment has many potential benefits," but, "developing in a surrogate environment requires that the developer be intimately familiar with the capabilities and limitations of both environments. Additionally, if there is not an automated transfer process, then the developer must manually map code from the surrogate environment over to the target simulation" [3]. This thesis seeks to make progress toward a more efficient method of transferring prototypes from a Unity3d environment to COMBATXXI.

III. REVIEW OF WXXI FRAMEWORK

A. COMBAT SIMULATION ENGINE

WXXI was designed to be similar to COMBATXXI with a focus on creating a simulation environment that allowed for easy and rapid AI prototyping. WXXI implements a discrete event system to process combat related events. The motivation for a discrete event system is partly due to its similarity to the COMBATXXI system, but more so it helps provide the capability to do testing and experimentation. Rapid prototyping relies on an ability to test functionality through multiple replications in the shortest amount of time. The current Unity environment is not optimized for this need. With this in mind, WXXI provides a discrete event system that provides both a testing and experimentation mode.

B. REPRESENTING TERRAIN AS A NAVIGATION GRAPH

The foundation for terrain representation and pathfinding in this research is the A* Pathfinding Project [12], which is a paid software package purchased from the Unity Asset Store. The A* Pathfinding Project comes with source code and provides the user with formal documentation and plenty of flexibility to customize tools to meet the user's needs. Because WXXI was developed for rapid prototyping, it requires a tailored terrain representation that holds specific information for planning and prototyping. An annotated mobility graph is a custom graph implementation that fits within the pathfinding project's architecture and holds additional information that supports the planning modules within WXXI [2]. In this case, VisibilityGraph is a custom graph type derived from the NavGraph class of the pathfinding project, which holds specific additional information such as risk and visibility [2]. NavGraph is the base class within the pathfinding project from which all graph types derive.

VisibilityGraph uses custom three-dimensional triangles called TerrainTriangleMeshNodes to represent nodes. Similar to the graph implementation, the nodes must also be annotated to support the planning modules within WXXI. In this case, TerrainTriangleMeshNodes extends TriangleMeshNode, which is an organic node implementation within the A* Pathfinding Project and is annotated with unit and entity

visibility information. All succeeding discussions regarding “nodes” and “graphs” can be assumed to be referring to TerrainTriangleMeshNodes and VisibilityGraph, respectively.

In addition to holding specific information for the planning modules, the standard node visualization within the pathfinding project was also modified to aid with troubleshooting and allow visualization of the additional information stored in each node. Specifically, each node is annotated with a node penalty that represents the risk value of that node as well as information on the visibility of that node by enemy entities. Simply, the risk value indicates the likelihood a friendly unit is expected to take casualties at that node location. More discussion on risk value can be found in Chapter V. Nodes that can be observed by any number of enemy entities are outlined in red. Additionally, Nodes with a risk value penalty greater than zero are marked with a yellow volume in the center of the node. The larger the yellow volume, the greater the killing rate at that location. See Figure 3 for a visual representation of a VisibilityGraph.

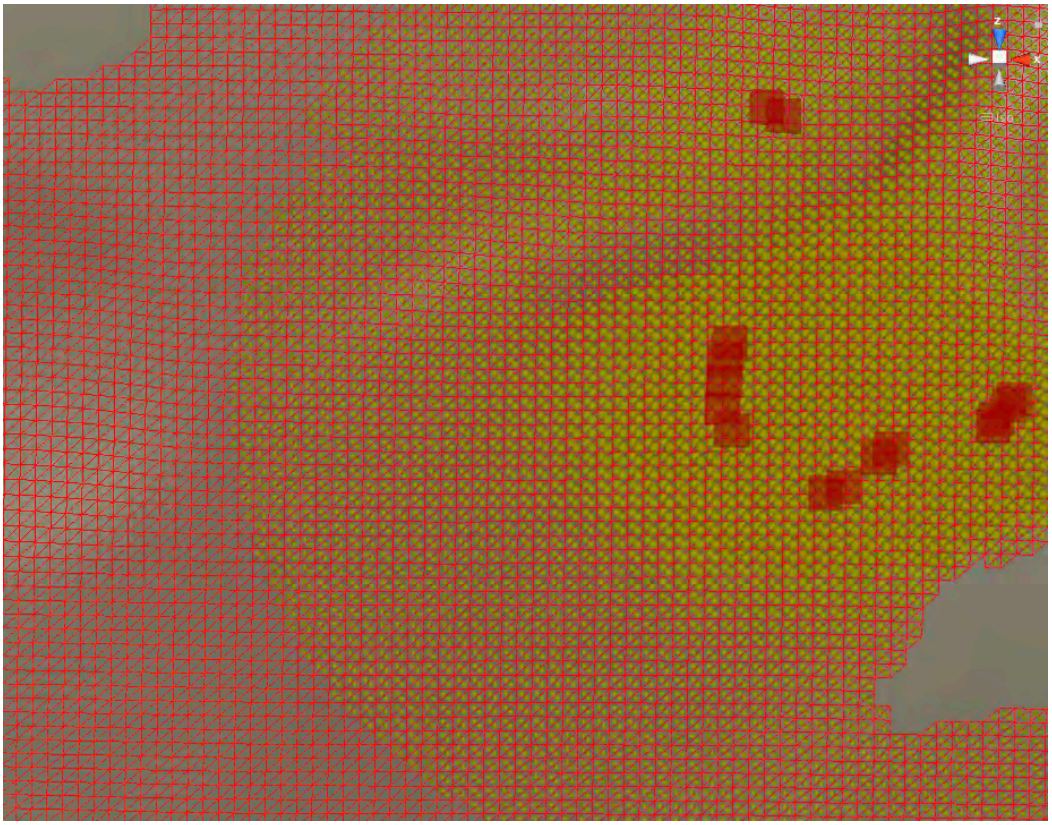


Figure 3. VisibilityGraph: Red Squares Represent the Location of Enemy Entities

C. PLANNING FRAMEWORK

The WXXI Automated Battle Planning System (ABPS) consists of three major components; planning input, planning data, and plan generator, which interact together and with the simulation environment to produce an executable plan. Figure 4 shows the top-level framework for the ABPS. The planning data is meant to be an "enterprise database" that stores information relevant across multiple scenarios such as derived models for movement and combat engagement, task dictionaries, and tactics configuration dictionaries [2]. The planning input is drawn from both the simulation environment and the planning data and is comprised of information that relates to the friendly and enemy forces, map, tasking, and tactics. The plan generator is the major component that uses data from the planning input and planning data to produce a plan. The plan generator has the capability to produce mission plans and enhancement plans. The mission planner produces an

executable maneuver plan based on a task by reasoning about the friendly units, enemy units, and any data provided as input. The mission planner simply produces a plan that achieves a given task, but does not claim to produce a doctrinal plan, or even a plan that works well [2]. Enhancement plans but modify or improve an existing plan. The enhancement planner must be provided a mission plan and assumes that the provided plan already achieves the task but will use an objective function and some known tactical knowledge to enhance the plan [2].

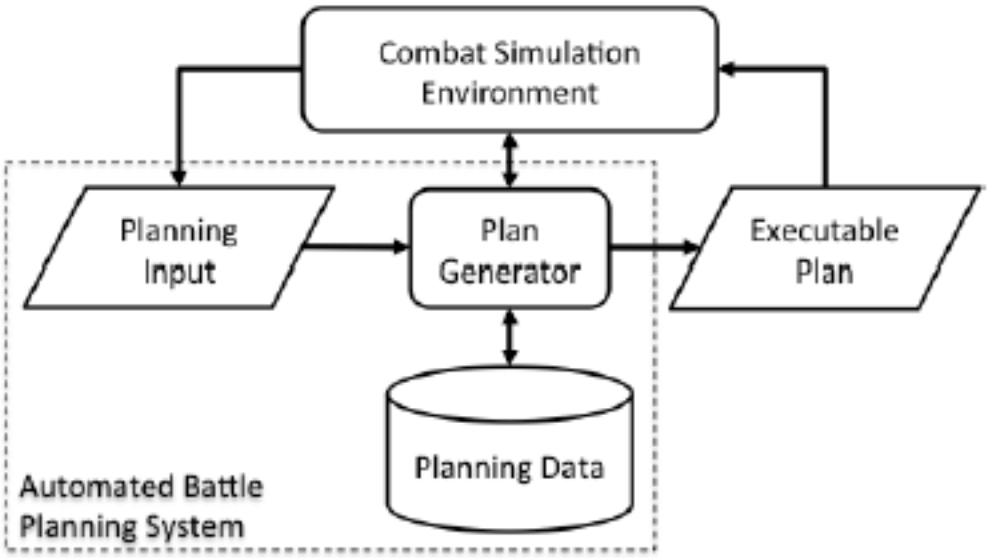


Figure 4. WOMBATXXI Top-level Planning Framework.
Source: [2].

The focus of prototyping in WOMBATXXI was on the development of a fire support enhancement planner that could improve an existing mission plan by reducing the expected number of casualties a unit would incur by executing that plan. Because the focus was on the fire support enhancement planner, the mission planner is not fully automated and requires user input to develop a plan. One of the required user inputs is tactical control measures which essentially allows the user to define the key decisions of the maneuver plan. In this thesis, we attempt a tactical position selection approach which will automate

the selection of tactical control measures that can be accepted by the mission planner and used to task subordinate units. This will generally create a doctrinal maneuver plan that could be improved through the use of an enhancement planner. The mission planner in WXXI assumes an enemy defensive position with an assaulting friendly force. We assume the same and focus the tactical position selection system on a platoon-sized friendly force. Of note, the tactical position selection system in this work does not determine the timing of friendly maneuver on routes or between tactical control measures. This decomposition is left to the WOMBATXXI mission and enhancement planners.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TACTICAL POSITION SELECTION

A. TACTICAL POSITION SELECTION DEFINED

In the commercial game industry, position location is highly scrutinized because it is one of the most visible aspects of the game. A game user can easily get frustrated with dumb AI behavior that interferes with the game experience. In the military community, poor position selection results in unrealistic agent behavior, and in real-world operation, leads to casualties and poor operation outcomes. Tactical position selection is simply the process of choosing movement locations based on a set of desirable criteria [13]. A common approach used in Tactical Position Selection systems is utility theory, which is the process of assessing a set of locations with respect to the desired criteria and choosing the best position from the set [14]. It follows that in this type of system, terrain must be represented in a manner that allows for assessment and queries of multiple locations.

B. TACTICAL POSITION SELECTION IN COMMERCIAL GAME DEVELOPMENT

Tactical position selection in commercial game development is usually used for individual entities as opposed to units, and the set of position locations available for selection is generally limited to a 10–15meter radius around the agent, which represents the immediate area that entity can move to or act within. Popular examples of this type of system are demonstrated in Killzone's position evaluation system or Crysis 2's Tactical Point System (TPS) [15]. In both cases, sample positions are generated at runtime. More recent research in this area has used a tactical position selection approach to identify defensive firing and covering positions in an urban environment using a "multi-objective optimization" technique that aims to optimize tactical requirements simultaneously [16]. In the urban example, similar to the common tactical position selection used in commercial gaming that limits the spatial environment for consideration to a small radius around the agent, the spatial environment is limited to a small 2D area representing a building floor plan. This thesis will take a more holistic approach and present a solution to tactical position selection for multiple units with the aid of a visibility graph (See Chapter III) to

represent a much larger part of the terrain rather than a small subsection generated at runtime for selection.

C. COMMERCIAL BEST PRACTICES

1. Utility Theory

As discussed, many tactical position selection systems use some implementation of utility theory to evaluate and choose locations. The concept behind utility theory is that every action is scored at once, ranked, and a high scoring action is selected [14]. The scoring is based on a set of factors chosen by the developer that are believed to affect a specific decision. The value assigned each factor represents its desirability or usefulness, and by extension, the total value assigned to a possible action based on those factors represents the overall desirability of selecting that action. Utility theory provides a more nuanced evaluation criteria than a simple threshold approach. All succeeding discussions referring to "scores" and "scoring" can be regarded as the value assigned to a factor, and the act of assigning value to a specific factor based on information from the simulation environment, respectively.

2. Choosing and Scoring Factors by Example

Let's say a couple is trying to decide what restaurant to eat at on a date night. They may consider price, service, travel distance, etc., as their decision factors. Too high a price or poor service may drive the couple to choose a different restaurant based on their budget or expectation of service. One of the challenges of a utility approach is comparing factors that are not easily comparable. In this example, some conversion between money (price), time (approximate time it takes for the food to be brought to the table), and distance (travel) must be made to properly understand the value or desirability of a specific restaurant. Consider assigning a dollar amount to time and travel distance. Every 5 minutes that the couple waits for their food is worth one dollar and every mile traveled is worth one dollar. Now the couple is able to compare the value of each restaurant over directly comparable criteria. A restaurant that will cost 100 dollars is 10 miles away and takes an average of 20 minutes to deliver food would have a value or utility of 105 dollars, which would be compared to every other restaurant that the couple is considering. The couple could also

take additional steps to be more precise in their conversion by dividing the distance traveled by the miles per gallon the car gets while driving, then multiplying by the price of gas to get an accurate representation of the cost of travel to a specific restaurant.

This type of system raises an important question regarding how many factors is too many. In general, there is no perfectly right answer to this question. Factor selection and scoring is somewhat of an art. Some games such as chess have a more direct and obvious relationship between factors and actions where piece mobility, board locations, and the safety of the king translate nicely into action selection [15]. Other domains such as first-person shooter games have a more complicated environment and a larger number of potential factors that may affect a large number of actions (not even considering different characters may have different goals). In general, large amounts of testing and experimentation across various environments and situations should be conducted until adequate agent behavior is achieved.

3. Normalizing Scores

Because scores are compared to each other to come up with a final action selection it is important that scores are converted to the same scoring scale across the system [14]. In addition, scores can often be combined to create other meaningful scores. In the example from above, if the couple lived in a busy city with traffic problems, the couple may combine a travel time and a travel distance score to produce the overall travel score for that particular restaurant. Normalized scores are a convenient and easy way to add consistency and simplicity to a scoring system, which can be complicated depending on the number of decision factors [14]. In this case, we use normalized to mean ranging in value from 0 to 1. Normalized scores are easy to average, can combine easily, and provide a transparent method for debugging. The developer automatically knows that a 1 is the most desirable and 0 is the least desirable for any given factor or cumulative score. Normalized scores combined with the power of weighting provide a robust system to evaluate a wide range of factors and actions.

4. Weighting Factors

While a given action may have a variety of factors that affect its utility, not all factors are created equal, and in some circumstances, specific factors may be more important than others. Weighting is one technique used to emphasize some factors over others. In commercial game development, weighting is a technique used to add personality to characters or AI agents. For example, consider an AI agent that chooses actions solely based on two factors, desire to attack based on proximity to an enemy entity, and desire to survive based on proximity to covered positions. At any given state, both factors will be scored and weighted, and the highest utility action will be chosen. Weighting the desire to attack higher than desire to survive adds an aggressive personality to the AI agent, which results in a bias to attack versus moving to cover. Conversely, weighting the desire to survive higher will result in agent behavior that favors cover over conflict and resembles a survivalist personality.

D. ADVANTAGES OF USING UNITY3D FOR TACTICAL POSITION SELECTION

The output of a tactical position selection system can be visually striking if it looks unreasonable or does not adhere somewhat to human behavior or reasoning. Consider a simulation agent that is being engaged and losing health quickly but does not attempt to utilize the cover right in front of him. This behavior is obviously unreasonable and frustrating to users. The development of a realistic tactical position selection system requires constant fine-tuning of factors, scores, and weights to achieve desirable results that are not outside the norm of what a human may decide without any supporting system in place [13]. This comes with an obvious need for a robust visual debugging system that can aid in the tuning and provide flexibility in development. Unity3D, in addition to the full spectrum of infrastructure it provides including documentation, support community, and video tutorials, also provides a robust visual debugging capability. The visual debugging capability allows the developer to see the real-world imported terrain as well as the graph that represents the terrain. The output of the tactical position selection system can be viewed and debugged on the real-world terrain and graph representation without interfering with any data that may be used by other systems in the WOMBATXXI

framework. Lastly, Unity3D has a built in Asset Store that makes available extensions and software packages including the A* Pathfinding Project [12], which provides the necessary assets used to build the tactical position selection system used in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

V. USING TERRAIN ANALYSIS TO ACHIEVE SMART TACTICAL POSITION SELECTION

A. TACTICAL CONTROL MEASURE SELECTION CRITERIA

As discussed in the development goals section, one of the main focuses of this work is on developing a planner that is realistic. The planner should not only output tactically sound and plausible maneuver plans, but the input data used by the planner to reason about the mission, units, and terrain should also be representative of actual information a platoon commander would receive from a standard operations order. Furthermore, realistic input data must be used to support similarly realistic planning factors that a platoon commander would consider when developing real-world maneuver plans. This section will detail the input data that is used throughout the maneuver planning system and then detail the threshold criteria and planning factors and their relevance to maneuver planning.

1. Input Data

The planning input data draws close similarity to the WOMBATXXI planning input discussed in Chapter III in that the information is drawn primarily from the simulation environment and user.

a. Terrain Representation

The primary input drawn from the simulation environment is the VisibilityGraph, which represents the real-world terrain as a mesh of triangle nodes. Each node represents a specific location on the terrain, which can then be annotated with information relevant to that node. Consistent with the original VisibilityGraph construction, each node is annotated with a risk value and observation information. Risk value represents the expected number of casualties a friendly unit will take by traversing that node. Observation information simply records the number of enemies that can observe a specific node out to a maximum observation distance. This is particularly important for a tactical position selection approach because it enables us to reason about locations that are outside the effective range of enemy fires without assuming that those positions are completely safe from the effects of other enemy supporting assets. An enemy unit in a static defensive position will certainly

have supporting fires that can affect a friendly unit outside of its direct fire range. Considering both risk and observation, we have a holistic view of the enemy effect on a particular location. Along the same lines, the cumulative risk and observation of a group of distinct position locations allows us to reason about larger areas, which may be occupied by larger units. This annotated information is used by both the pathfinding modules and planner to reason about the terrain.

b. Enemy

In order to develop maneuver plans in software or the real world, some information about the objective must be known. In real-world operations, this takes the form of known, suspected, or likely enemy positions in order of certainty with an enemy disposition. In this case, we assume a deliberate attack on an enemy in a static defensive position. For a deliberate attack, the level of certainty in the enemy location and disposition is high. The enemy information provided to the planner to reason about the tactical plan are:

(1) Center of Mass of the Enemy Location

It is unlikely that a platoon commander will have the exact location of enemy soldiers or even the location of subordinate units within the same defensive position. It is common to describe an enemy position as a single grid location or terrain feature such as a hill rather than an array of individual fighting holes. The planner is limited to a single center of mass position to replicate the ambiguity that a platoon commander will often face in real-world planning. Of note, the center of mass calculation does not include enemy observation or listening posts. It is assumed in offensive planning that the enemy will make some attempt to monitor avenues of approach or position posts for early warning, however, these posts are small, well concealed, and difficult to detect. Similarly, we do not assume any information about the location of enemy observation posts, but only about their observation over areas of the terrain.

(2) Average Orientation of the Enemy Unit

Similar to the discussion into the ambiguity of enemy locations in real-world planning, it is unlikely that a platoon commander will have specific information on enemy

fields of fire or an engagement area. An engagement area is commonly expressed as a general orientation in degrees that may cover the most prominent avenue of approach or the most advantageous area for the defending unit. With a general orientation, planners can reason about the front, flanks, and rear of an enemy defensive position without knowing the main engagement area or exact fields of fire of subordinate units. In this case, we express the orientation of the enemy force as a vector representing the average orientation of all its entities (excluding observation posts). This allows the planner to filter undesirable nodes prior to scoring any planning factors without considering any known risk values. It is important to note that the planner will still reason about specific risk of a position if it has not previously been filtered.

(3) Unit Size

While the enemy location and orientation can often be ambiguous, the enemy unit size if often assumed with a higher degree of certainty. In order to define the scale of the offensive operation and begin the planning process, a commander must know an approximate size of the enemy. We assume a known squad sized defensive element in a static defense.

c. Weighting

The maneuver planner is designed to be used as an editor tool prior to running the simulation, enabling the user to adjust weights and visualize the output prior to running multiple replication for analysis in a target simulation. Weights allow the user to set the level of importance for each planning factor and see the resulting output in the visualization of the maneuver plan. Weights are given a value between 0–1 and the sum of the weights for a given position must equal 1. This is a percentage approach to weighting where a greater weight value reflects a higher level of importance for that specific planning factor. The weight value is the percentage of that specific cost that will be considered in the total cost of a given position. Figure 5 demonstrates this construct through a trivial example. Assume that all costs in the table below are on the same scale. Notice that factor #2 has the highest cost, but since it is weighted at 0.2 or 20%, a smaller portion of that score is reflected in the total score compared to the smaller cost but higher weighted factor #1. It is

important to note that this weighting approach only works if all the planning factor costs are on the same scale. Otherwise, costs on a larger scoring scale will dominate the total cost regardless weights.

Total Cost for a Single Position

Decision Factor	Cost	Weight	Resulting Cost
Factor #1	15	0.5 (50%)	15 x 0.5 = 7.5
Factor #2	32	0.2 (20%)	32 x 0.2 = 6.4
Factor #3	24	0.3 (30%)	24 x 0.3 = 7.2
Sum	**71**	**1.0 (100%)**	**21.1 Total**

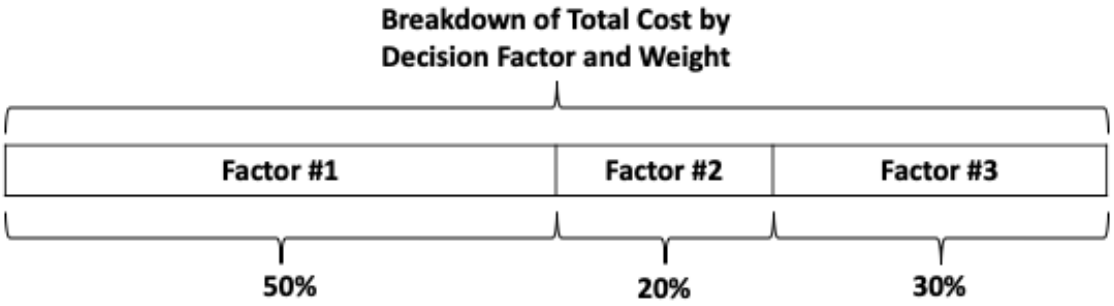


Figure 5. Example of a Total Cost Calculation

d. Tasking

Consistent with real-world planning, the user has the ability to set which units will be tasked as the main effort (ME), supporting effort #1 (SE1), and supporting effort #2 (SE2). The selection has implications on the resulting TCMs and routes that will be assigned as part of that specific unit's maneuver within the plan as a whole. However, this research is scoped to have friendly units with identical capabilities. This limits the amount of reasoning that must be done on the capabilities of the friendly force. While this may seem unrealistic because real-world units do have different capabilities based on attachments and special equipment, it is still consistent with real-world planning where a commander would first task a subordinate unit as the ME or one of the SEs, then attach

additional personnel or equipment to each subordinate unit based on its assigned tasks. Individual unit capabilities may be a valuable area of future work and a logical extension of this system.

All input data is either set by the user prior to running any system functionality or drawn from the combat simulation during the initialization of the maneuver planner. Further discussion of the data flow and planning steps is discussed in Chapter V, Section C. While the input data is tailored specifically for this planner, the data is consistent with common data used as input to the WOMBATXXI planning system and drawn from the same sources within the larger system framework.

2. Threshold and Sorting Criteria

It is important to distinguish between planning factors and the threshold criteria used to sort and filter out undesirable positions. Threshold criteria are similar to planning factors in that we assume the criteria represents important characteristics of a position that must be considered to developing a tactically sound plan. Threshold criteria are used for two purposes, first to filter out undesirable positions, and second to sort positions into like groups. To filter undesirable nodes, threshold criteria are simply used as a barrier to avoid scoring a large number of positions that we can easily throw out based on an undesirable characteristic of that position such as being too risky or too far from the objective. Moreover, planning factors are assigned a weighted score, which essentially allows us to rank the value of positions based on the planning factors. For filtering undesirable positions, we assume that anything below a certain threshold has no value whatsoever to the maneuver planner and should therefore be thrown out entirely. To sort, positions are evaluated over a set of criteria that allows us to determine what type of control measure that specific position should be considered for. It is computationally inefficient to score positions based on the planning factors for a certain type of control measure when it does not meet the basic characteristics that a position must have to be considered for that type of control measure.

We will next discuss the filtering of nodes based on thresholds to identify the positions that will be considered for attack positions, followed by the sorting of potential

assault positions. In this section, we assume an order of sorting, scoring, and selecting assault positions, followed by filtering, scoring, and selecting an attack position. The motivation for this ordering is discussed further in Chapter V, Section B.

a. Filtering for Attack Positions

The attack position is the last covered and concealed position before moving to the assault position. While a unit should be ready for contact at any point after crossing the LOD, the attack position is still considered relatively safe. One of the challenges of filtering out attack positions is that we want to filter out the undesirable positions, but we do not want the thresholds to be so strict that we throw out positions with potential or leave ourselves with very few positions to even consider for scoring. Conversely, if we cast too wide a net, we risk sacrificing computational efficiency by having to score large numbers of undesirable positions.

(1) Unwalkable

To begin filtering, we start by eliminating nodes that are unwalkable for entities. A position is tagged as unwalkable if it cannot be traversed by an entity because the gradient is too steep (greater than 30 degrees).[1] An unwalkable position may still be a neighbor to a walkable position that has met the minimum threshold criteria and will therefore be scored as a neighbor position to that walkable position in question. However, we do not consider unwalkable positions initially as attack positions because it is likely that surrounding positions will also be unwalkable based on similar terrain characteristics such as slope or obstacles.

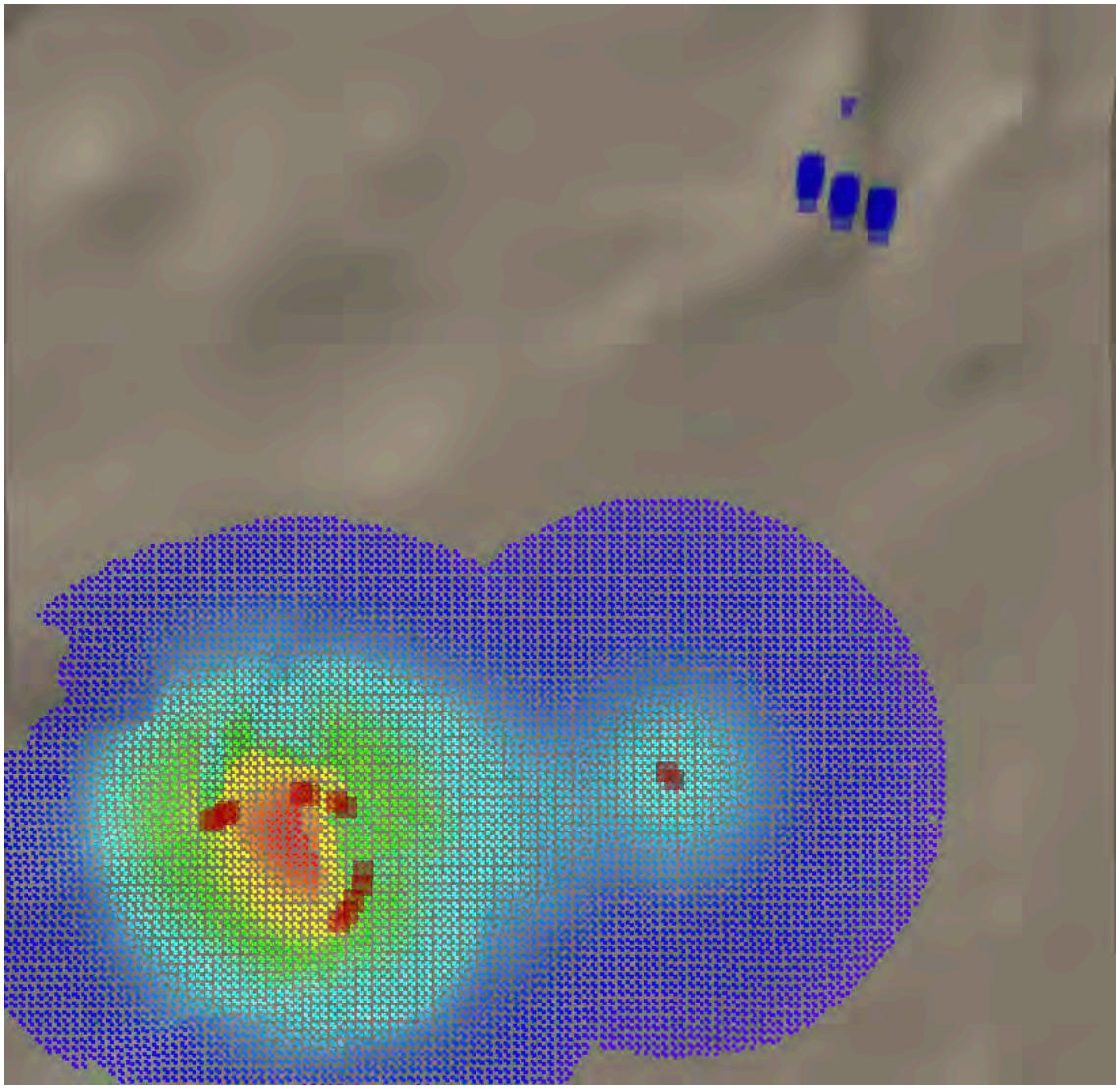
(2) Cover and Concealment

Because the attack position should be covered, we filter positions where we expect casualties by excluding positions with a risk value greater than zero. Figure 6 shows a depiction of the positions with a risk value greater than zero. The color scheme works from

[1] In general, this can be any trafficability limitation that prevents a simulated entity to cross, but in our case, this is limited to slope.

blue to red where blue is the lowest risk value (but still above zero), and red is the greatest risk value. Notice that all the positions with a risk value greater than zero are within the effective range of enemy direct fire weapons. Also, notice that the positions in the center of the enemy position have the highest risk values. This is reasonable because it is likely that all individual enemy positions will have a clear line of sight to the center of their own defensive position.

Selecting concealed positions requires a bit more fine-tuning. Ideally, an attack position would be completely concealed from enemy observation. However, this is unlikely unless the position in question is far enough away from the enemy position that it is outside the visual and observable range through basic optics. The farther away an attack position is, the greater distance, and potentially uncovered distance the unit must move to reach the assault position. In this case, we fine-tune the threshold criteria to exclude positions that have a greater number of enemy observers than the average for the operational area. This allows some positions that are unconcealed to be considered. Positions with a risk value above zero are filtered out prior to examining the observation of a position which provides certainty that although a position may be observed, it is still fully covered from the effects of enemy direct fire weapons. The risk of unconcealed positions comes from indirect supporting fires where an enemy can observe the friendly unit and call a firing battery to adjust fire on the friendly position. Realistically, a single observer could call for supporting fires on a friendly position. However, we still assume that positions with less observation are more valuable than positions with more observers.



The color scheme works from blue to red where blue is the lowest risk value (but still above zero), and red is the greatest risk value.

Figure 6. Visualization of Positions with a Risk Value Greater Than Zero

(3) Distances

Lastly, we filter positions based on characteristics of their distance from other key areas. First, we ensure that the position in question is outside of the range of enemy direct fire weapons. Because the risk value of each position considers the range of enemy direct fire weapons, no additional work is done to accomplish this.

Next, we want to ensure that a position in question does not add unreasonable distance to the plan by moving in the opposite direction of the enemy objective, or in a direction that gets the friendly unit no closer to the enemy objective. Some implementations may not restrict positions based on distance and allow positions scattered across the map to be considered. However, real-world planners are certainly concerned with the endurance and stamina of their subordinates and make every effort to reduce the amount of unnecessary distance covered. In this implementation, we do the same. A simple comparison of the main axes of approach serves as a starting point to eliminating undesirable positions. Figure 7 visually depicts the axes and thresholds described below. As a threshold, we eliminate nodes that have a greater straight-line distance from the position in question to the enemy objective than from the unit's starting position to the enemy position. This would mean that a unit would be traveling farther to get to the objective from the position in question than simply going directly to the objective from the unit's starting position without ever going to the intermediate position in question. Not to mention the additional distanced traveled to get to the position in question. Using this criterion ensures that each move gets us closer to the enemy.

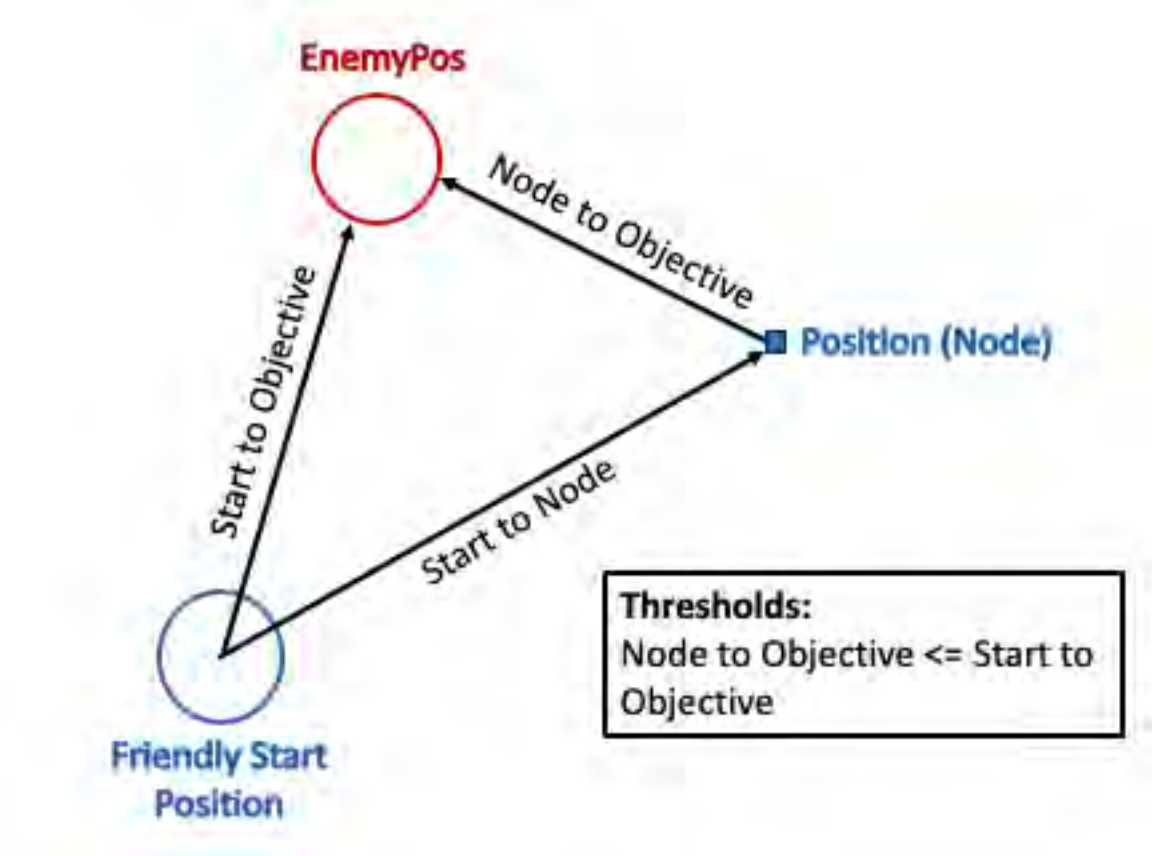


Figure 7. Visual Depiction of the Major Movement Axes a Unit Will Take to the Enemy Objective

After filtering positions that add unnecessary distance, we want to again filter out positions that do not get the friendly unit closer to the enemy objective. The ratio depicted in Figure 8 gives us a measure of how close a position gets the friendly unit to its objective.

$$\frac{(Distance\ From\ Start\ to\ Objective) - (Distance\ From\ Node\ to\ Objective)}{(Distance\ From\ Start\ to\ Node)}$$

Figure 8. Ratio Representing How Close a Position Gets a Unit to Its Objective. Source: [13].

It is important to note that this ratio is calculated with straight line values, not an A* tactical path calculation. The planning factors will certainly take the actual path into account when scoring positions that have met the minimum thresholds but calculating

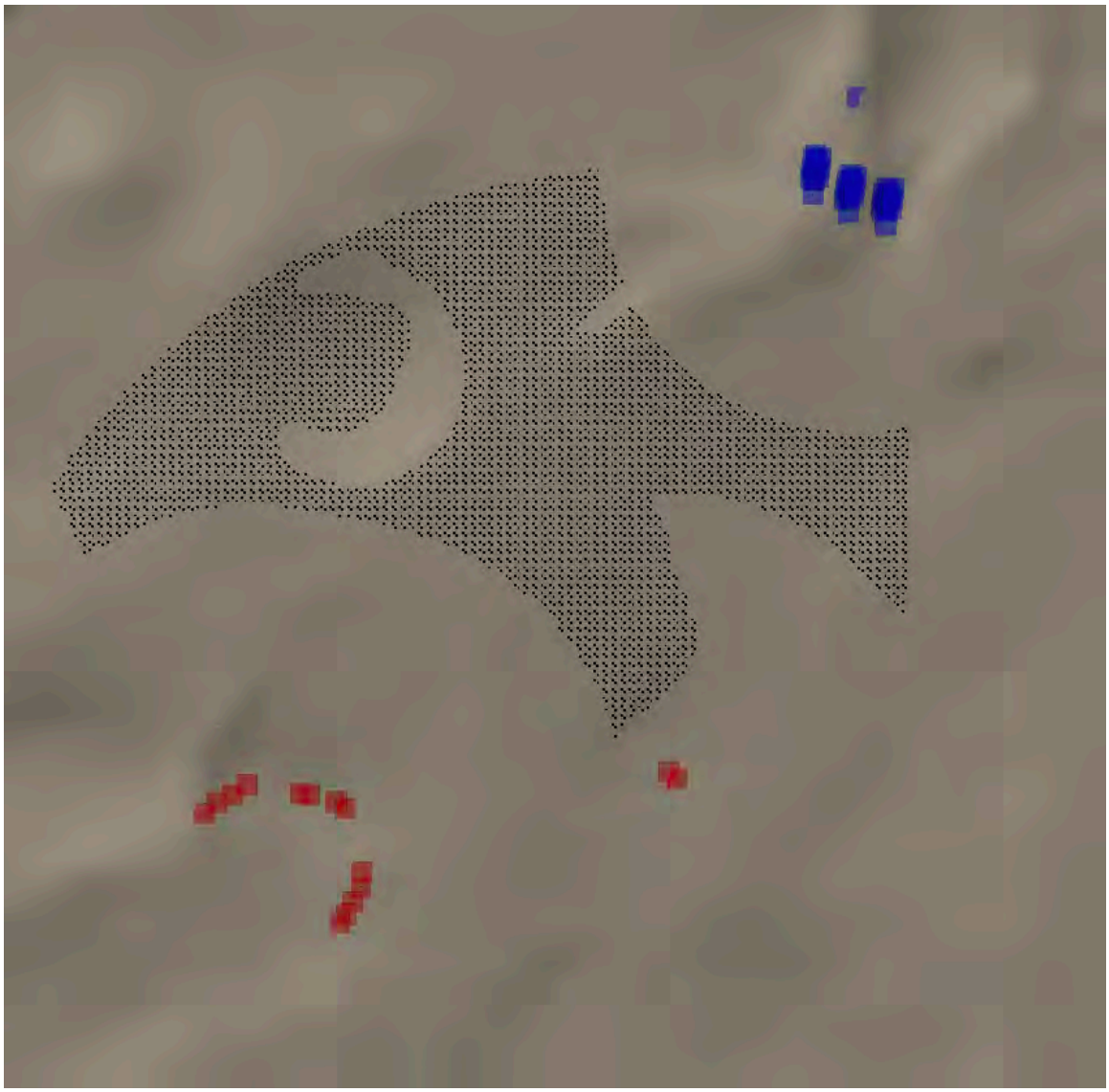
tactical paths as a filtering technique adds an unnecessary computational burden that can otherwise be solved with a simple ratio. A ratio of 1 represents a direct straight line to the objective where for every meter moved, the friendly unit gets 1 meter closer to the objective. A ratio of 0.5 means for every 2 meters moved, the unit gets 1 meter closer to the objective. While the precise threshold value can be adjusted by the user, we use a value of 0.5 as the threshold value.

After all the attack position filtering criteria has been executed and unnecessary positions have been discarded, the remaining positions are stored in a list and will eventually be scored based on the planning factors for an attack position. Figure 9 below depicts the attack position filtering criteria in pseudo code and Figure 10 depicts the resulting positions that are stored after filtering.

```
filterAttackPos() {
     potentialAttackPosition = List<Graphnode>
     graph = VisibilityGraph
     enemyPos = center of enemy position
     friendlyPos = starting position of the friendly unit
     observers = # of enemies that can observe node

     foreach (node in the graph)
          nodePos = position of node on graph
          if (node is walkable)
               if (observers < average observers across the graph)
                    if (distance from start to node <= start to goal)
                         if (distance from node to goal <= start to goal)
                              distance nodeToGoal = (enemyPos - nodePos).magnitude
                              distance startToGoal = (enemyPos - friendlyPos).magnitude
                              distance startToNode = (nodePos - friendlyPos).magnitude
                              ratio = (startToGoal - nodeToGoal) / (startToNode)
                              if (ratio >= 0.5)
                                   Add node to potentialAttackPosition list
}
```

Figure 9. Pseudo Code: Attack Position Filtering Criteria



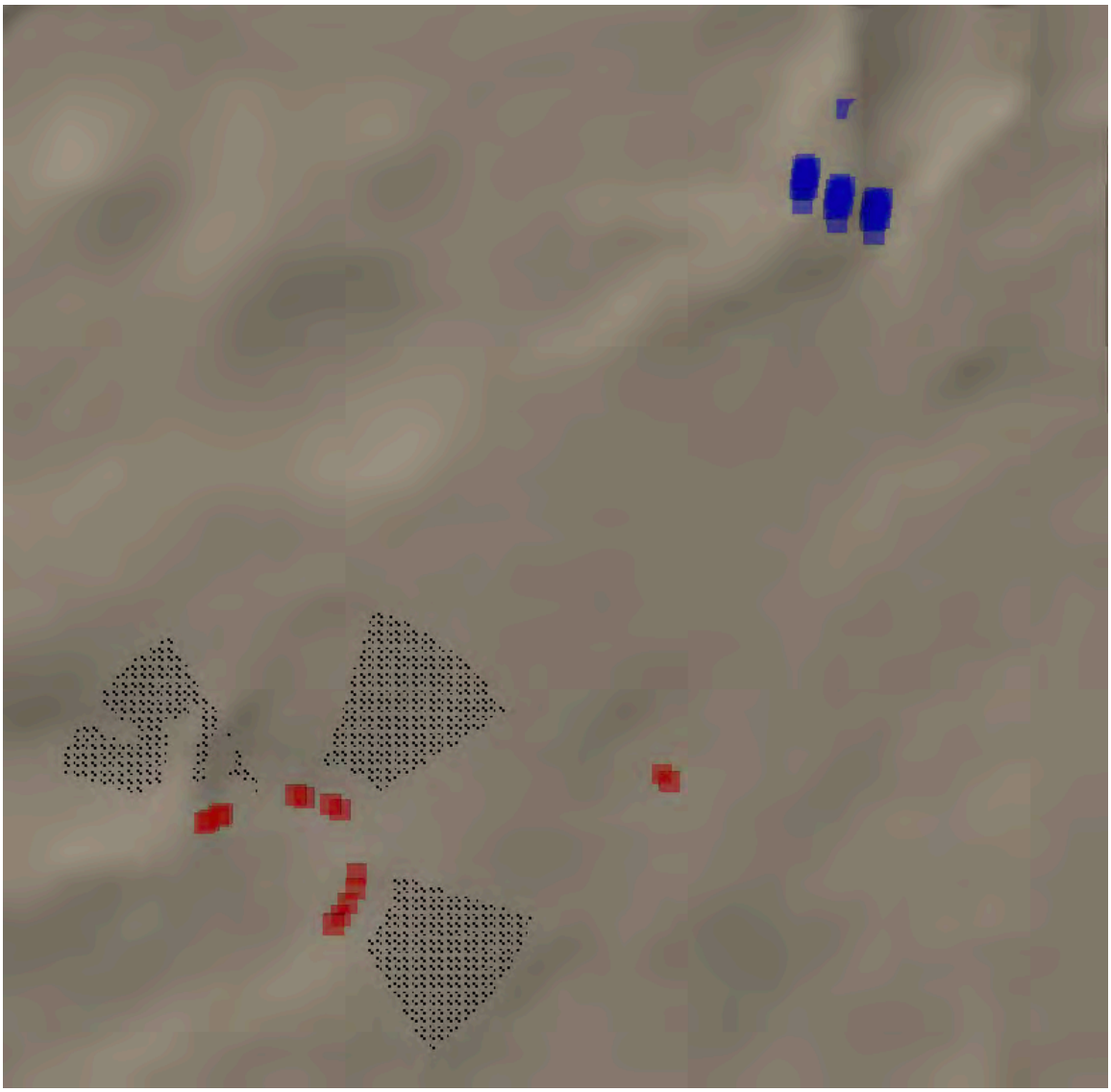
Positions are colored black because they are unscored.

Figure 10. Visualization of Positions That Have Met the Minimum Thresholds to be Considered for an Attack Position

b. Sorting and Categorizing Assault Positions

Sorting assault positions is much less nuanced than filter attack positions because the real-world criteria for an assault position is strict particularly because we assume that immediately after leaving the assault position, which is the last covered position before the objective, the friendly unit will be under fire from enemy direct fire weapons. In this case, we consider dividing the objective into sectors representing the front, left flank, and right

flank of the enemy, and bound those sectors with a reasonable minimum and maximum assault distance. The minimum and maximum assault distance is a user-defined threshold and is likely to change based on the terrain, time of attack, avenue of approach, and many other factors. We use default values of 75 and 300 meters, respectively. The 75 meters was selected because it is outside the range that a truly athletic human could throw a grenade. The 300 meters was selected because that is as close as a unit can get to the objective, with small exceptions, before indirect supporting fires must cease firing. The sectors are defined based solely on the single position representing the enemy position, and the average orientation of the enemy force. Both gathered from the mandatory input data discussed in Chapter V, Section A. Three positions will eventually be selected from the defined assault position sectors; the ME assault position, SE1 assault position, and SE2 assault position. The ME assault position will be selected first, followed by SE1 and SE2. Additional filtering is done after the ME assault position is selected and before the SE1 and SE2 positions are selected to ensure that the potential supporting effort positions are not at risk of friendly fire from the ME assault position. This is accomplished by removing positions from the assault position sectors that are within a 15-degree cone to the front or rear of the ME assault position. Positions within the 15-degree cone to the front of the ME assault position are at risk of being the victim of friendly fire, positions in the cone to the rear put the ME assault position at risk of being the victim of friendly fire from that position. Figure 11 depicts an example of the defined assault position sectors.



The positions are colored black because they are unscored. Notice that the sector to the left of the enemy position has gaps. These gaps are nodes that have been filtered out because they are unwalkable due to a steep incline greater than 30 degrees.

Figure 11. Depiction of the Assault Position Sectors

3. Planning Factors

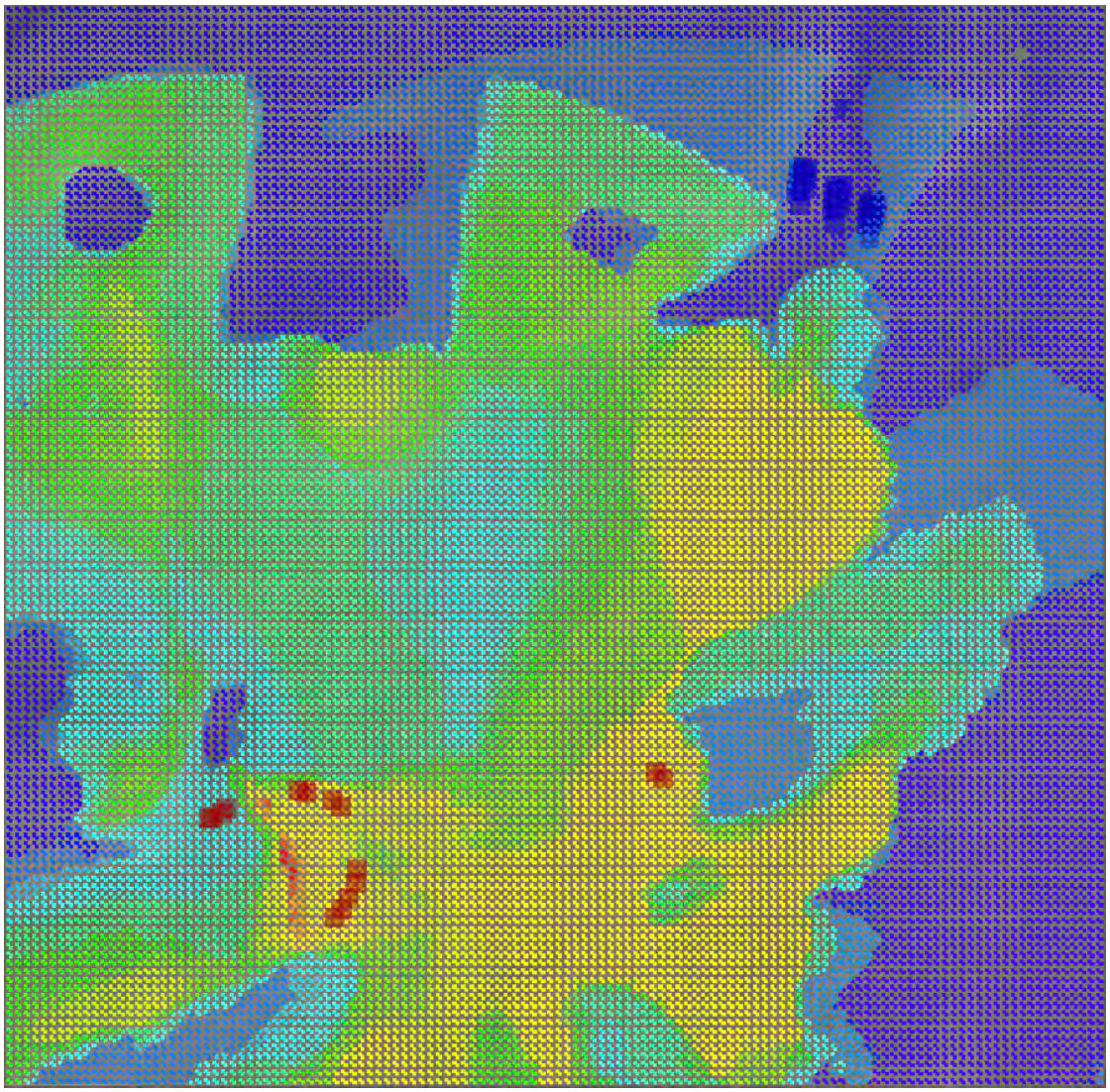
The core of this system is the selection and scoring of planning factors. This section will detail the planning factors used by the planner to reason about the terrain, mission, and enemy and friendly force. In order to develop a system that is representative of real-world planning, we must also reason about the same factors that human planners' reason about.

This is where a doctrinal understanding of maneuver planning coupled with operational experience is vital. An automated maneuver planner simply cannot reason about factors that a human planner would not consider without sacrificing the tactical significance of the resulting plans.

a. Attack Position Planning Factors

(1) Observer Cost

The observer cost is a measure of the concealment of a position. A fully concealed position will have an observer cost of zero and a fully unconcealed position will have an observer cost equal to the total number of enemy entities present on the enemy position plus the enemy entities in any observation posts hidden throughout the map. This means that every enemy entity can observe the position in question. It is obvious that a fully concealed position is valuable, and a fully unconcealed position is undesirable. However, these are both edge cases that are not prevalent in most areas of the terrain. In this case, the observer cost allows us to rank positions based on their measure of concealment. The lower the observer cost, the more concealed a position is. Consistent with the nodeObserver implementation in WOMBATXXI, the observer cost is calculated using raycasting where a ray is cast from the height of an enemy entity to each node vertex on the VisibilityGraph within the maximum visibility distance set by the user [2]. If the ray does not hit any intervening terrain, then every node that includes that vertex is labeled as observable [2]. Figure 12 shows a visual depiction of the raw observer cost. The large cluster of red and blue squares are the friendly and enemy position. The visibility maximum distance is set to 1500 meters. The color pattern works from blue to red with blue being the lowest observer cost (most concealed) and red being the highest observer cost (most unconcealed). Note that the nodes with the highest observer cost are located in the center of the enemy position. This is reasonable because it is likely that the center of the enemy position will be in view of all its individual defensive positions.



The maximum visibility distance is set to 1500 meters. The color pattern works from blue to red, with blue being the lowest observer cost (most concealed) and red being the highest observer cost (most unconcealed).

Figure 12. Visualization of the Raw Observer Cost

(2) Neighbors Cost

The neighbors cost is a measure of the cost of occupying a large area. When considering an individual entity, it may be reasonable to consider only individual nodes to represent the location where that entity is located. However, when reasoning about larger units, we must consider the full area that a unit will occupy which includes many nodes. In this case, we assume that the entire friendly unit will occupy a single attack position. While

some situations may call for additional attack positions, especially if a friendly unit has attachments, it is common practice that a platoon-sized unit occupy a single attack position. Final preparations before the assault are conducted in the attack position, which is more easily accomplished if the unit is co-located. Additionally, any changes to the plan will be more easily communicated face-to-face rather than over line-of-sight radios, which are vulnerable to rough terrain. An extension of this system may consider additional attack position for subordinate units or attachments such as machine gun squads.

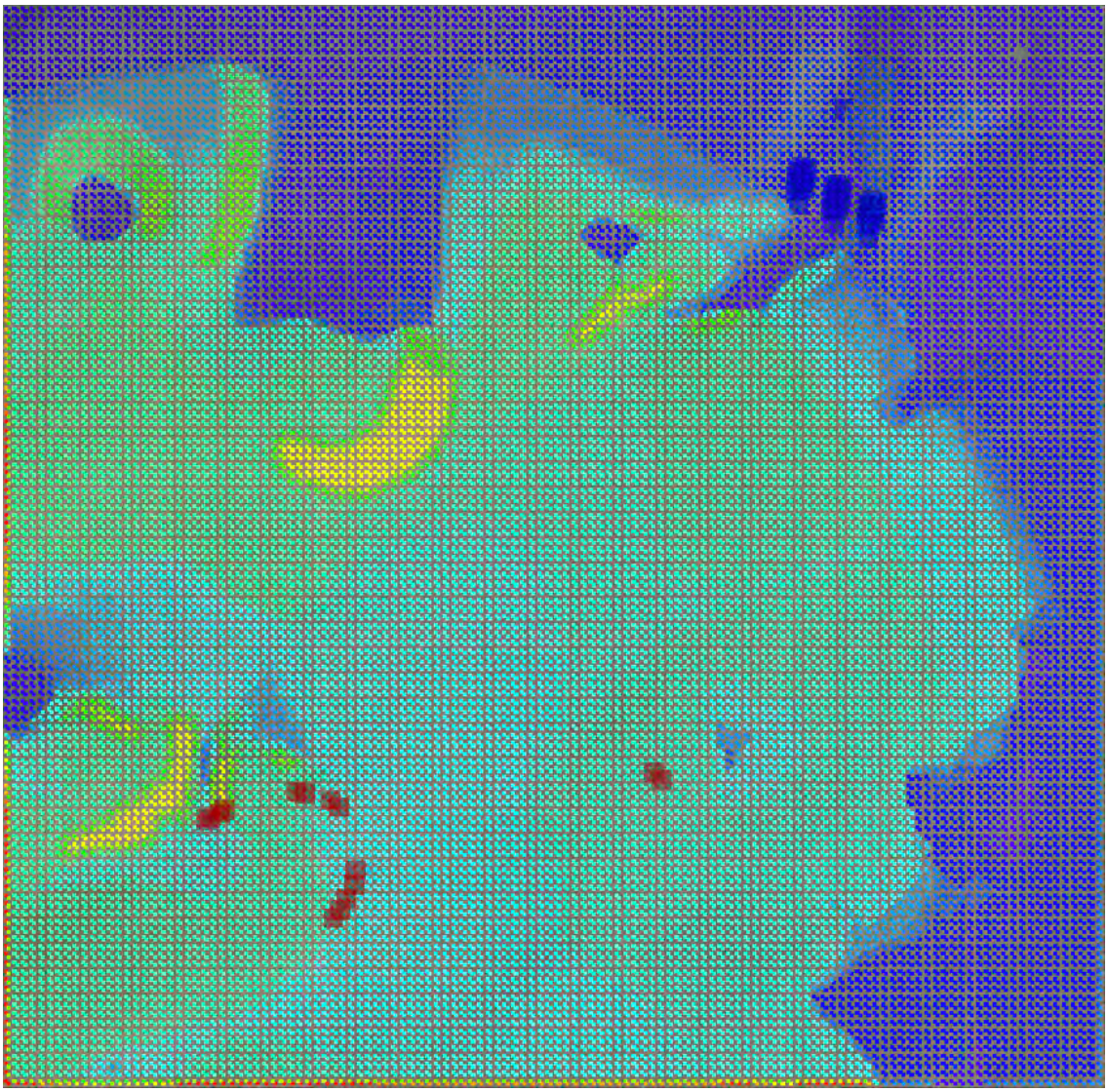
In order to properly reason about a positions suitability to contain an entire platoon, we have to reason about the neighboring nodes to the position in question. To demonstrate, consider a node that is in a small depression on the terrain, which is concealed from the enemy, but surrounded by open ground that is completely visible from the enemy position. If only the node in the depression is considered, that position will look suitable because it is concealed, but a large unit cannot all occupy a small depression and will have to occupy the open ground as well. The neighbors cost for an attack position is calculated by summing the observer cost of all neighboring nodes within the radius of a platoon-sized unit and then averaging over the expected number of nodes within that radius. Positions closer to the edge of the map may not have the full number of expected nodes within the platoon radius. In this situation, we penalize the position in question by calculating how many nodes are missing from the expected number of nodes and penalizing each of those positions with the maximum observer cost. The expected number of nodes is calculated by dividing the surface area of a platoon-sized element by the size of a node on the VisibilityGraph. We set the platoon radius to be 50 meters, assuming the platoon will take a standard 360-degree security posture in the attack position. Figure 13 shows the pseudo code of a neighbors cost calculation and Figure 14 shows the visualization of a neighbors cost layout for a platoon radius of 50 meters. The color scheme works from blue to red, with blue being the lowest neighbors cost and red being the highest neighbors cost. Notice that the red high scoring nodes are located around the edges. This is because we penalized positions that did not meet the expected number of nodes, which represent the positions that cannot hold an entire platoon.

```
getNeighborsCost() {
     platoonRadius = 50
     expectedNumberNodes = platoonArea / nodeSize
     foreach potential attackPos node
          Loop through every other node on map
               if (node is within platoonRadius of attack position node)
                    add node to list of neighbor nodes

          foreach list of neighboring nodes
               if (numberOfNeighboringNodes < expectedNumberNodes)
                    difference = expectedNumberNodes - numberOfNeighboringNodes
                    penalize the difference with maximum observer cost
               if (neighboring node is !Walkable)
                    penalize with maximum observer cost
               assign all other neighboring nodes their true observer cost
               total = sum of all observer costs of neighboring nodes

     totalCost for potential attackPos node = total / expectedNumberNodes
     return dictionary of neighbors cost for every potential attackPos node
}
```

Figure 13. Pseudo Code: Neighbor Cost Calculation



Color scheme works from blue to red with blue being the lowest neighbors cost and red being the highest neighbors cost. Notice that the red high scoring nodes are located around the edges. This is because we penalized positions that did not meet the expected number of nodes.

Figure 14. Visualization of the Neighbors Cost for a Platoon Radius of 50

(3) Directness Cost

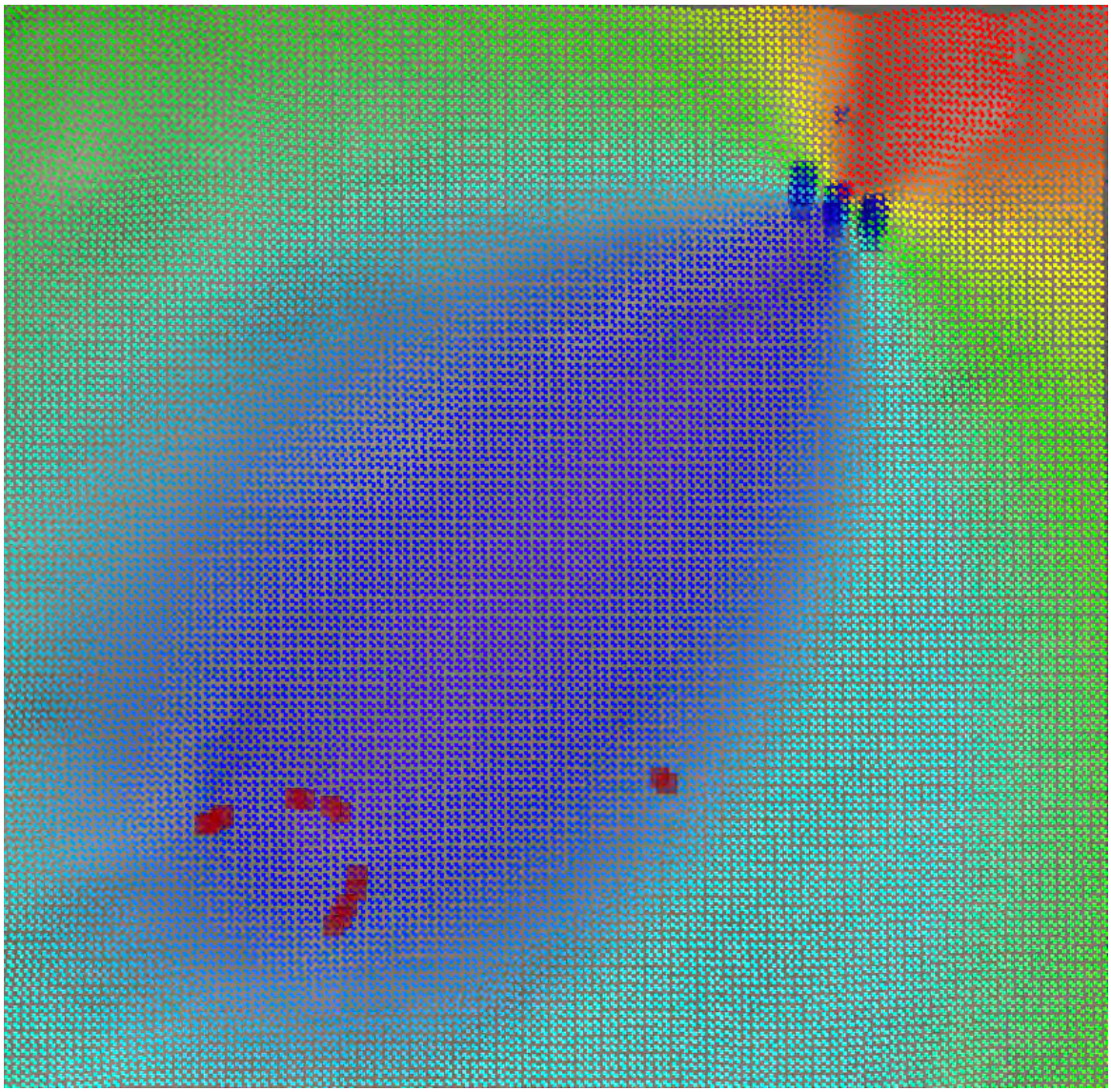
The directness cost is the same ratio depicted in Figure 8, which is a measure of how close a position gets a unit to its goal [13]. When filtering, we use this as a measure of how much closer the position in question gets the friendly unit to the enemy position. In this case, because we do not move from the attack position directly to the enemy objective,

we use this as a ratio of how much closer the attack position gets the friendly unit to its assault position. While Chapter V, Section C discusses this further, it should be mentioned that the scoring system was designed so that the lowest cost nodes are the most desirable. The directness cost is not consistent with this design because a higher directness cost means a more direct path to the goal, which is desirable. Because only positive directness ratios greater than 0.5 meet the minimum threshold criteria to be considered for an attack position, we simply take the inverse of each position's directness cost so that the lowest cost node is now the most direct. Figure 15 depicts a directness cost calculation in pseudo code and Figure 16 shows an example of the directness cost for each node where the starting location is the friendly position identified by the larger blue rectangles, and the goal location is the enemy position identified by the larger red rectangles.

```
getDirectnessCost() {
     assltPos = ME assault position
     friendlyPos = starting position of the friendly unit
     foreach potential attackPos node
          nodePos = position of node on terrain
          distance nodeToEnemy = (assltPos - nodePos).magnitude
          distance friendlyToNode = (nodePos - friendlyPos).magnitude
          distance friendlyToEnemy = (assltPos - friendlyPos).magnitude

          directnessCost = 1 / [(freindlyToEnemy - nodeToEnemy) / (friendlyToNode)]
     return dictionary with directness cost for every potential attackPos node
}
```

Figure 15. Pseudo Code: Directness Cost Calculation



The starting location is the friendly position identified by the larger blue rectangles, and the goal location is the enemy position identified by the larger red rectangles. The color scheme works from blue to red, where blue is the lowest cost or most direct to the goal, and red is the highest cost or least direct to the goal. Notice that the worst positions (red) are the positions that move in the opposite direction of the enemy position.

Figure 16. Directness Cost Visualization

(4) Path Cost

One of the most important real-world planning factors is routes or avenues of approach. Planners not only consider the travel distance, but also how long it will take to traverse a specific route and the risk of observation, indirect fire, and direct fire from the

enemy position. Planners seek to balance these considerations in order to select routes that give the friendly unit the highest probability of making it to the objective while sustaining the minimum number of casualties in the process or maintaining some element of surprise or deception in the process. In some situations, a planner may trade some security along a route for a faster or shorter route to the objective. In other situations, a planner may favor a longer path that provides better cover and concealment or is more likely to maintain the element of surprise. Ideally, a route minimizes the distance traveled while maximizing cover, concealment, and the element of surprise. The intent of combining the risk value and travel distance into the cost metric is to capture this tradeoff when paths are generated. However, it is unlikely that the tactical scenario will provide the ideal route. After all, the enemy also seeks to maximize their observation over the terrain, cover the most prominent avenues of approach with direct fire weapons, deploy observation posts for early warning, and select defensive positions that are not vulnerable to complete surprise attacks.

To construct a path, we use a traditional tactical pathfinding approach were both the distance between nodes and the penalty of each node in the path are included in the cost [17]. The node penalty assigned to each node in the graph is the risk value or expected casualties associated with that node. The total cost of a path is the sum of the traversal distance and risk values for each node in the path. Consistent with the WOMBATXXI implementation, each node penalty is multiplied by a modifiable penalty weight, which serves as a scaling factor and allows the user to determine the relative importance of risk in the overall path cost [2]. In this case, we use the A* Pathfinding Project's ABPath implementation to calculate tactical paths between two points [12]. An ABPath is the most general path type provided by the A* Pathfinding Project [12]. In general, an ABPath simply constructs the least cost path between two points. Often this is the shortest path. However, the addition of node penalties allows the construct to function as traditional tactical pathfinding [17]. The ABPath constructor takes the start and end position as arguments and uses the A* search algorithm, with node penalties included, to calculate a tactical path between the start and end positions.

Each potential attack position's path cost is calculated by constructing two separate tactical paths and combining their total scores. The first is constructed from the friendly

unit's starting position to the potential attack position in question. The second is constructed from the potential attack position to an already selected assault position. The sum of the distance between each node in the path and each node's individual penalty (risk value) are stored as the path cost for each potential attack position in question. Figure 17 shows a path cost calculation for a potential attack position. Further discussion on the ordering of filtering and scoring is discussed in Chapter V, Section C.

```
getPathCost() { //for an attack position
     startPos = friendly starting position
     node = potential attack position node
     asssltPos = already selected assault position

     foreach node
          path #1 = tactical path from startPos to node
          foreach path #1
               sum distance of each leg in path #1
               sum individual risk values of each node in path #1
          totalCost path #1 = sum of distance + sum of risk

          path #2 = tactical path from node to asssltPos
          foreach path #2
               sum distance of each leg in path #2
               sum individual risk values of each node in path #2
          totalCost path #2 = sum of distance + sum of risk

     totalCost for node = totalCost path #1 + totalCost path #2
     return dictionary with path cost for every potential attack position

}
```

Figure 17. Pseudo Code: Attack Position Path Cost

As a path is being constructed, the A* search calculates its distance using a straight-line between the center of each triangular node. This brings about an unnatural zigzag behavior which takes longer to traverse (See Figure 18). In this case, after a path is constructed, it is postprocessed using a modifier to smooth its appearance and create a more direct path (See Figure 18). The A* Pathfinding Project includes various modifiers that can smooth paths based on the type of navigation graph and environment being used [12]. WOMBATXXI uses two modifiers called RadiusModifier and FunnelModifier [2]. This implementation uses a single RaycastModifier which specifically reduces zigzagging [12]. Instead of moving directly from waypoint to waypoint, raycasting is used to identify the furthest waypoint the entity can travel to in a straight line [18]. The intermediate waypoints are removed from the path resulting in a smoother path [12].

Figure 18. Visual Depiction of a Path Without Modifiers on the Left, and With a RaycastModifier Applied on the Right

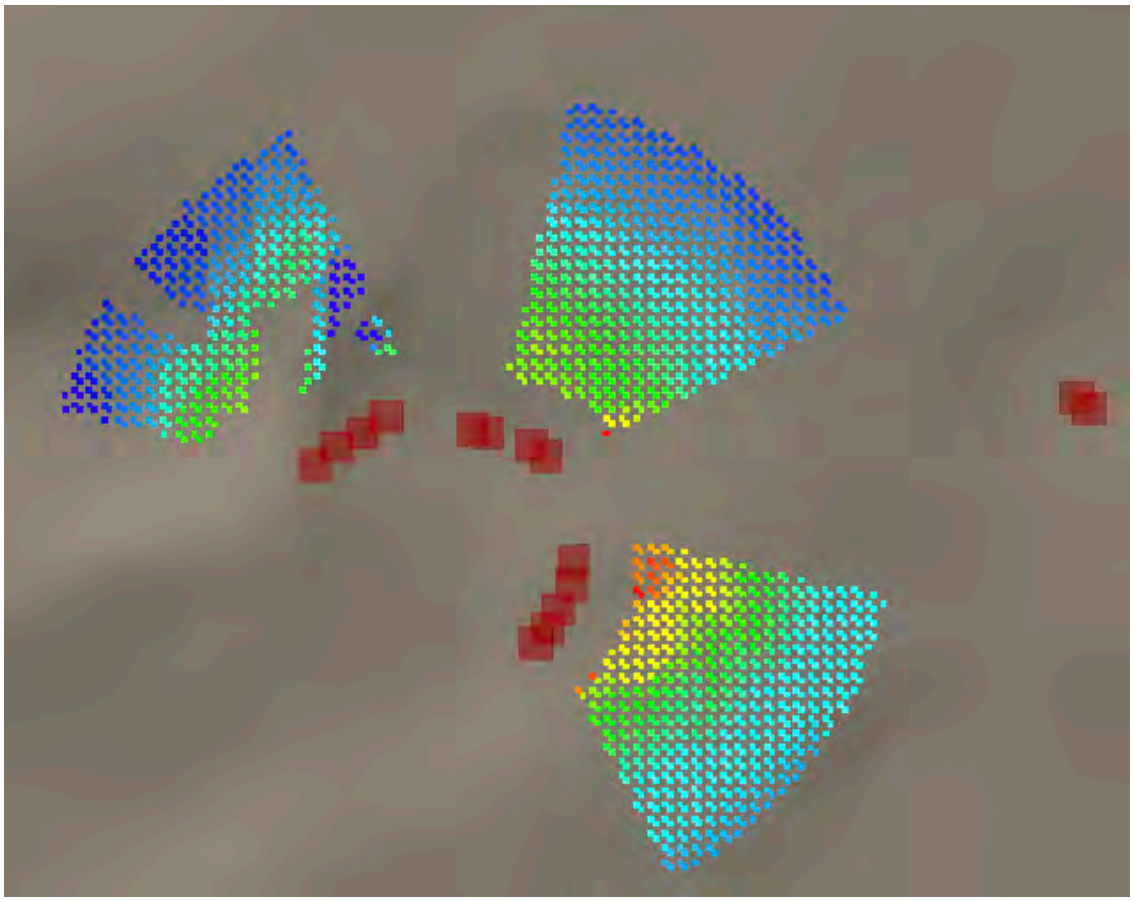
b. Assault Position Planning Factors

The assault position is the last covered and concealed position before the objective. It is expected that the friendly force will be observed or directly engaged once leaving the assault position. As discussed, the potential assault positions are sorted into three sectors

that comprise the left flank, right flank, and front of the enemy position. While the positions are sorted into three sectors, the positions are considered one collection of potential assault positions and scored as such. Therefore, position scores are relative to the entire collection, not just the positions in the same sector. This in effect allows the planner to determine the type of attack (frontal, flanking left, or flanking right) by selecting the lowest scoring position from the entire collection rather than being constrained to a single sector.

(1) Risk Value

The most basic factor that any planner would consider is the risk associated with occupying a particular position. Particularly the risk of casualties from enemy direct fire weapons. A good assault position should provide cover from the effects of enemy direct fire weapons. In this case, we use the risk value or expected number of casualties annotated in each position as its risk value. This value is stored as the position penalty during the generation of the VisibilityGraph, so no additional work is done to calculate this value. Additionally, because all costs are normalized during the scoring process, the penalty weight is still included in the risk value cost. Figure 19 depicts the assault position sectors with only the risk value cost.



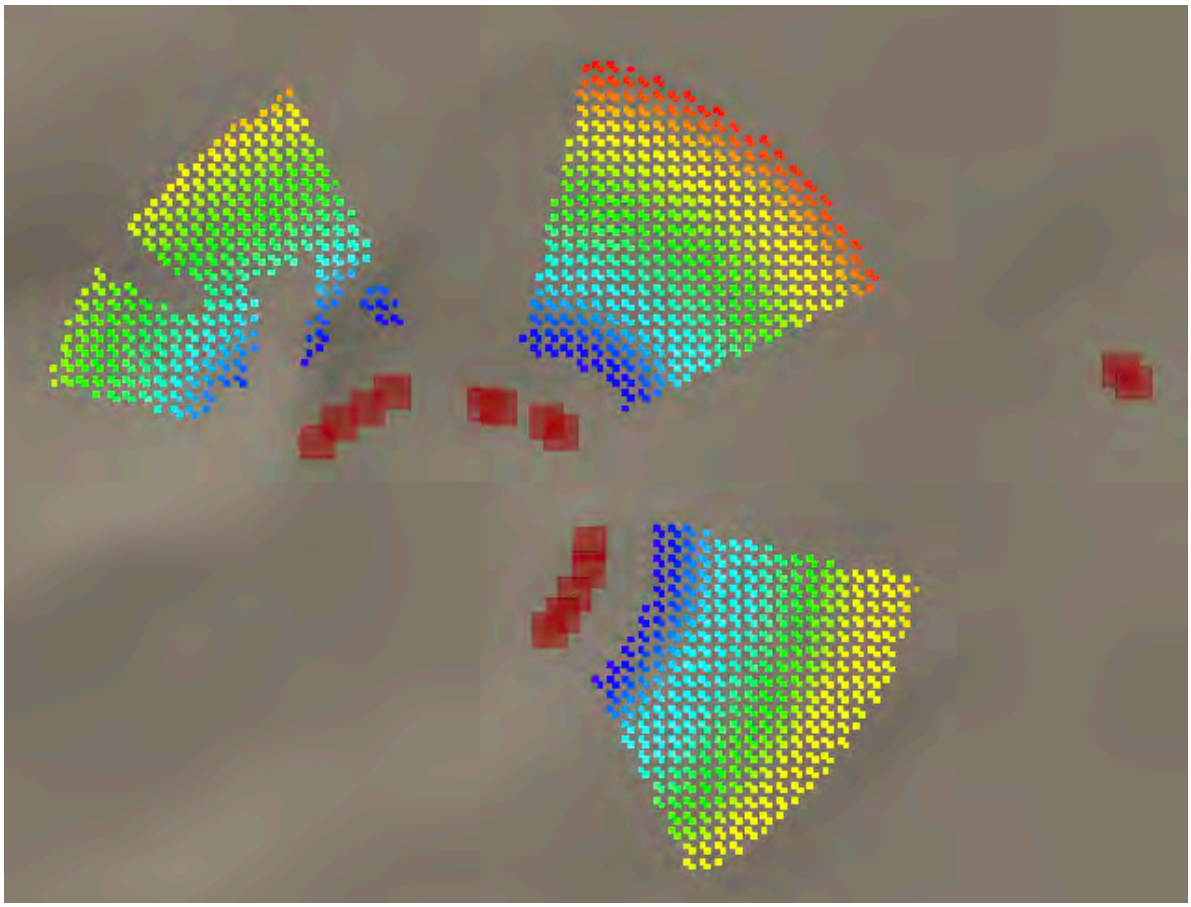
The color scheme works from blue to red, where blue is the lowest cost or least risk and red is the highest cost or most risk.

Figure 19. Visual Depiction of the Risk Value Cost

(2) Assault Distance Cost

As discussed in the assault position filtering section, each sector is bounded by a minimum and maximum assault distance. This cost is simply calculated by measuring the straight-line distance from the potential assault position to the enemy closest to that position. The assault position is the last covered and concealed position before the objective and friendly entities will likely be under enemy fire as soon as they leave the assault position. With this in mind, we assume first that friendly entities will take the most direct route from the assault position to the enemy objective, and second that positions closer to the enemy are more favorable because they reduce the distance a unit will need to travel while under fire. Note that this factor does not include any information about the risk value

of the position in question or along the route to the objective. We leave that consideration for other planning factors. Figure 20 depicts the assault position sectors with only the assault distance cost.



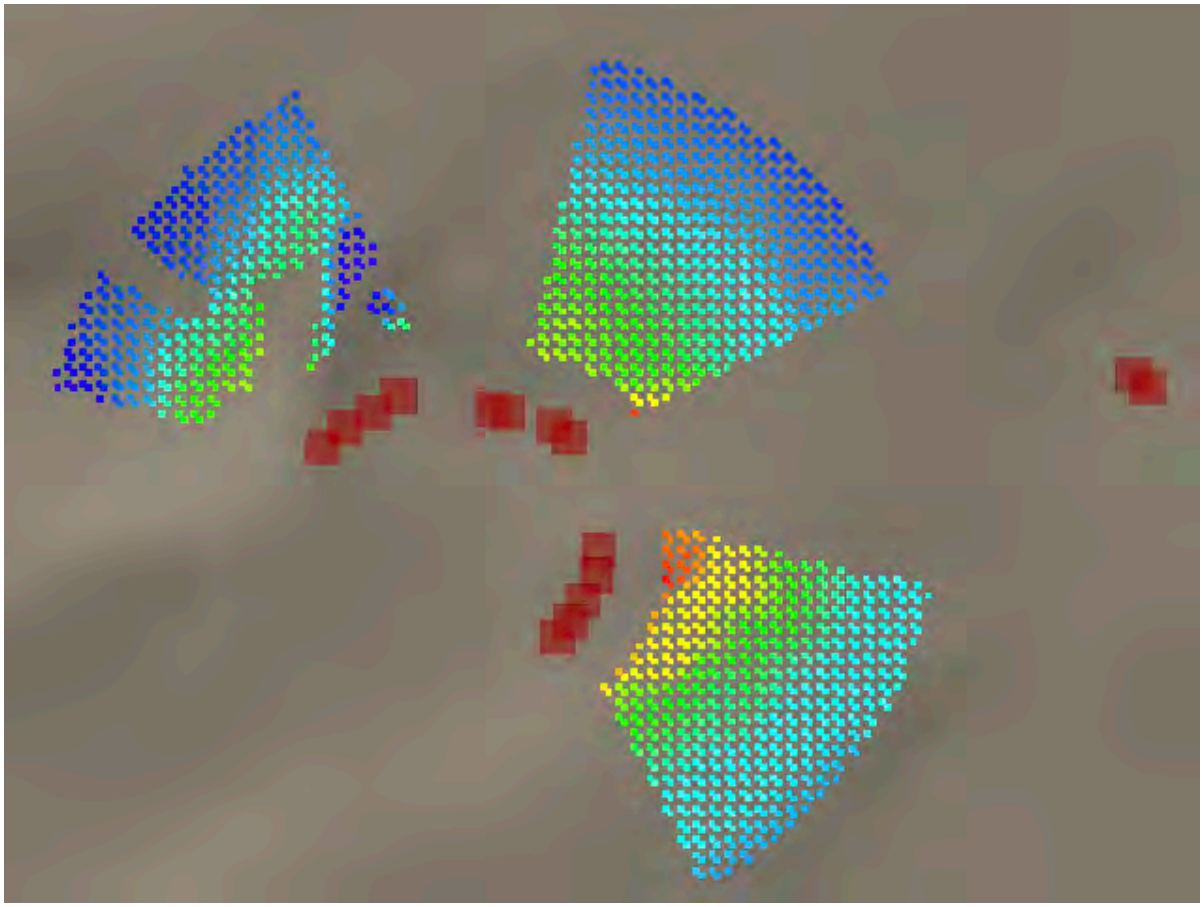
The color scheme works from blue to red, where blue is the lowest cost or closest to the nearest enemy and red is the highest cost or farthest from the nearest enemy.

Figure 20. Visual Depiction of the Assault Distance Cost

(3) Neighbors Cost

The assault position neighbors cost follows the same reasoning and implementation as the attack position neighbors cost with a few important differences. First, we assume that only a squad-sized element will occupy an assault position. Therefore, the neighbors cost is calculated over a squad sized area rather than a platoon sized area. This is consistent with doctrinal platoon maneuver where one squad will be tasked as the ME with the

responsibility of assaulting the objective, and the remaining two squads will be assigned as SEs and will occupy separate positions, but still within the assault position sectors. Additionally, because the assault position is well within range of the enemy's direct fire weapons, the risk value is considered instead of the observation cost. The assault position should be a covered position, but the observation only gives us information on its concealment. Moreover, by the time a unit gets into its assault position, it is likely that the enemy will have some intelligence that an assault is happening. Figure 21 depicts the assault position sectors with only the assault distance cost.



The color scheme works from blue to red, where blue is the lowest cost or most permissive squad area and red is the highest cost or least permissive squad area.

Figure 21. Visual Depiction of the Neighbors Cost

(4) Combined Arms Cost

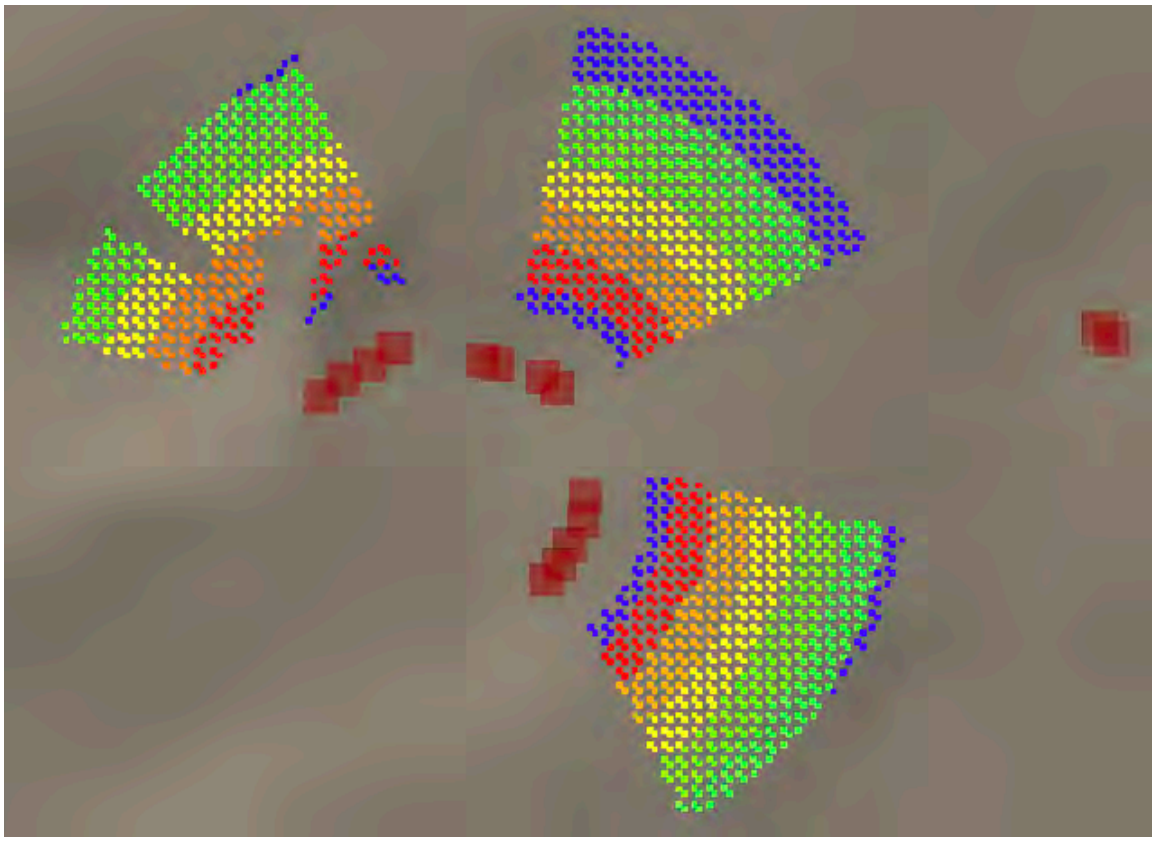
The combined arms cost is a measure of risk from organic enemy indirect fire weapons such as a grenade launcher. A defensive position is likely to have dead space that the enemy entities cannot target with their direct fire weapons. However, dead space will likely be covered with an organic indirect fire weapon. The effective range of most organic indirect fire weapons is between 100 and 350 meters. While an experienced soldier can effectively engage targets out to the maximum effective range, we assume that accuracy decreases as range increases. Ranges that are outside of the effective range are assumed to have a probability of hit of zero. The combined arms cost is calculated as probability of hitting a target at that range. The range is calculated as the distance between the position in question and the closest enemy to that position. In this implementation, the probability of hit is scaled linearly as range increases; however, an extension of this system may seek to find accurate probability of hit data for a specific organic indirect fire weapon. Note that this cost does not include the effects of fragmentation from an indirect fire weapon because each node is assigned this cost separately. Figure 22 and 23 show the pseudo code of the combined arms cost calculation and the depiction of the assault position sectors with only the combined arms cost.

```
getCombinedArmsCost() {
     minRange = 100
     maxRange = 350
     assltPos = potential assault position node
     closestEn = closest enemy to the node in question
     Phit = probability of hit

     foreach assltPos
          distance = (assltPos - closestEn).magnitude
          if (distance <= maxRange && distance >= minRange)
               combinedArmsCost = Phit(distance)
          else
               combinedArmsCost = 0

     return dictionary with combinedArmsCost for each assltPos
}
```

Figure 22. Pseudo Code: Combined Arms Cost

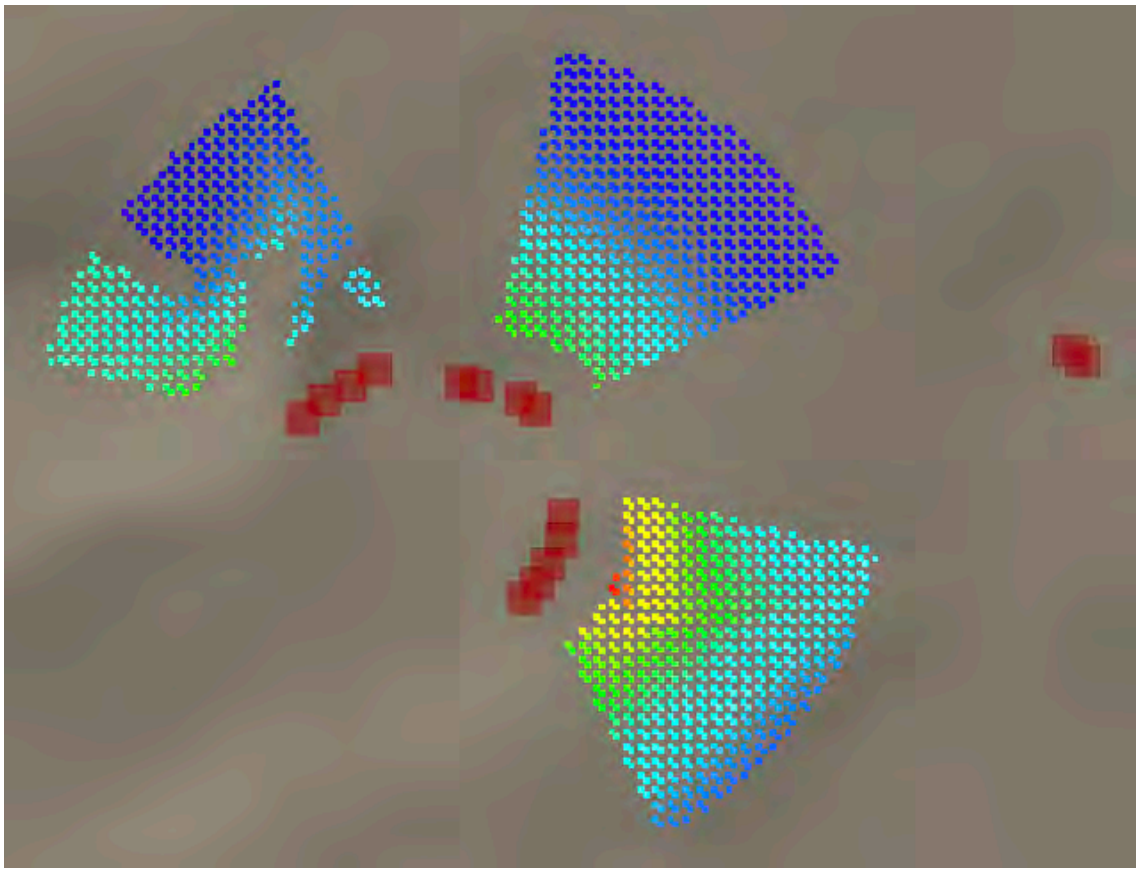


The color scheme works from blue to red, where blue is the lowest cost or outside the effective range and red is the highest cost or in most in danger of organic indirect fire weapons. Notice that positions outside the effective range are blue.

Figure 23. Visual Depiction of the Combined Arms Cost

(5) Path Cost

The path cost for an assault position follows the same logic as the path cost for a potential attack position with one difference. The path cost for an assault position is calculated as an estimate of the actual path cost. Assault positions are selected prior to selecting an attack position. Therefore, the actual cost of moving from the attack position to the assault position is unknown. Instead, we estimate a cost by constructing a tactical path from the friendly units starting position to the potential assault position. The actual path cost of moving from the attack position to the assault position is captured in the second path that is calculated when scoring the path cost of the attack positions. (See Figure 17.) Figure 24 depicts the assault position path cost estimate.



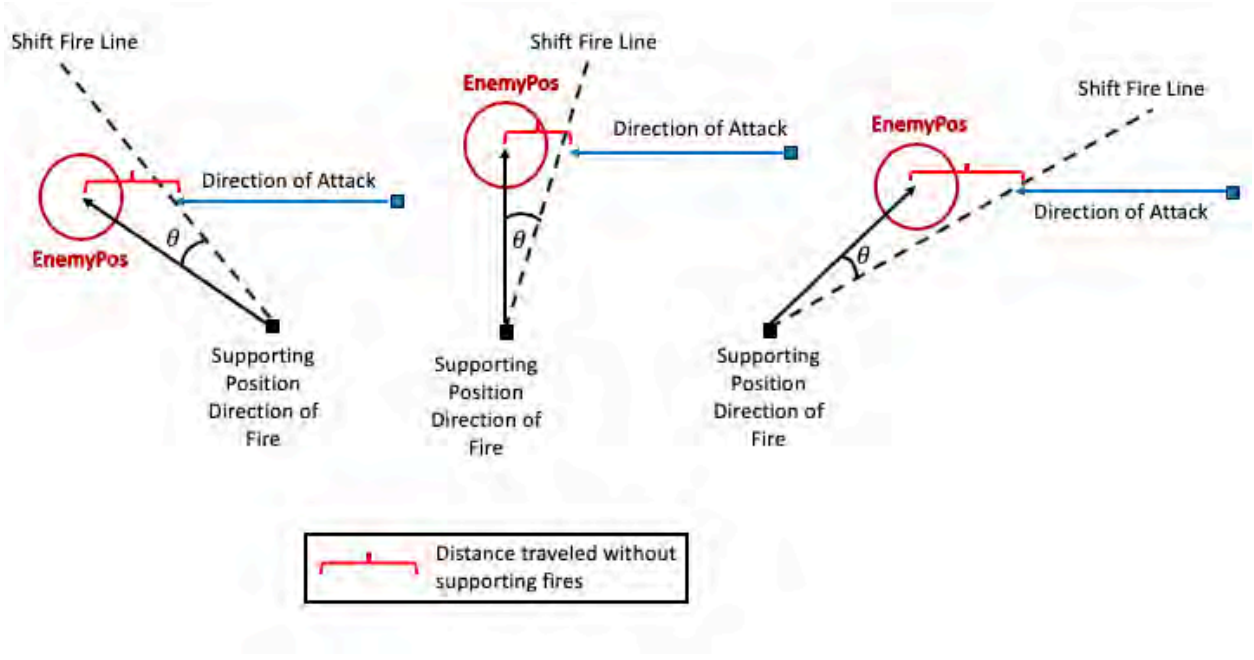
The color scheme works from blue to red, where blue is the lowest path cost and red is the highest path cost.

Figure 24. Visual Depiction of the Path Cost

(6) Supporting Fires Cost

The supporting fires cost is a measure of how well a potential position can support the ME assault on the objective. Typically, in a platoon assault, one squad will be assigned as the main effort with the responsibility of assaulting the objective, while the other two squads will be in support. This means they are responsible for providing suppressive fires for the ME and maneuvering on the objective after the ME has cleared the objective. A good supporting position will allow supporting fires to suppress the objective for the maximum amount of time before having to shift and cease fires. This comes down to simple geometry. A supporting unit must shift its fires away from the assaulting unit when its fires are within 15 degrees of the most forward entity in the assaulting unit. If a supporting unit takes too wide or too shallow an angle, their fires will need to be shifted and ceased

prematurely. A 90-degree offset from the assaulting unit provides the maximum amount of suppression time for the assaulting unit. Figure 25 depicts these three scenarios. Theta for all three situations is a doctrinal 15-degree offset from the direction of fire. The scenarios on the left and right depict a supporting position that has taken too shallow an angle and too wide an angle from the assaulting unit, respectively, which results in premature shifting of fires. The middle scenario depicts an ideal 90-degree offset from the assaulting unit.

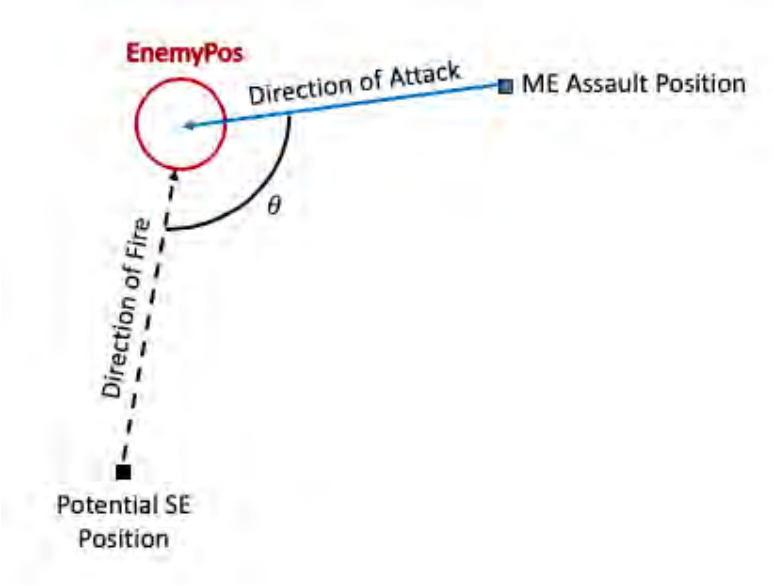


The scenarios on the left and right depict a supporting position that has taken too shallow an angle and too wide an angle from the assaulting unit, respectively, which results in premature shifting of fires. The middle scenario depicts an ideal 90-degree offset from the assaulting unit.

Figure 25. Depiction of Three Separate Supporting Fires Scenarios

In this case, the supporting arms cost is assigned to the remaining positions in the assault position sectors only after the ME assault position has been selected. To calculate the supporting arms cost, the angle between the direction of attack vector and the direction of fire vector is calculated (See Figure 26). The particular calculation returns the smaller of the two signed angles between the two vectors. We do not discriminate based on the sign, so the absolute value is taken. Angles between 45 and 135 degrees are the most

desirable for supporting fires and are assigned a cost of zero. All other angles are simply assigned a cost equal to their measured angle. Since the low scores are better than high scores, this naturally favors shallow angles between the direction of attack and direction of fire. A shallow angle means the supporting position will be closer to the assaulting unit, which potentially allows for better communication and line of sight between the two positions. A wide angle means the positions are farther apart and at a greater risk of faulty communication. Figure 27 shows pseudo code of a supporting arms cost calculation.



In this scenario, theta represents the smaller of the two angles measured from the direction of attack to the direction of fire.

Figure 26. Depiction of the Angle Measured for the Supporting Arms Cost

```
getSupportingArmsCost() {
     assltPos = node selected as the ME assault position
     enemyPos = center of the enemy position
     friendlyPos = starting position of the friendly unit
     node = potential supporting effort position

     foreach node
          vector from = assltPos - enemyPos
          vector to = node - enemyPos
          cost = signedAngle(from, to)
          supportingArmsCost = absValue(cost)

     return dictionary of supportingArmsCost for each node
}
```

Figure 27. Pseudo Code: Supporting Arms Cost Calculation

B. PRIORITIZING THE MAIN EFFORT

Now that we have discussed filtering, sorting, and the planning factors that will be used to score positions, we have all the pieces to begin scoring nodes and selecting the best positions for tactical control measures. Logically, the next step is to determine in what order we should filter, sort, and score the control measures. Real-world planners take a holistic approach where a plan is determined to be good, bad, or insignificant based on the consideration of all control measures together. However, planners must still prioritize the selection of some control measures over others. That determination is usually made with the main effort in mind because that is the unit conducting the assault on the objective. We take the same approach by selecting the ME assault position before any other control measure is considered. This position will be used as the basis to filter, sort, and store all other positions.

C. SCORING GRAPH NODES BASED ON THRESHOLD CRITERIA AND PLANNING FACTORS

We have chosen to select the ME assault position as our basis for filtering, sorting, and scoring all remaining positions. If a position meets the minimum criteria for an assault position (either ME or SE) or attack position, it is stored in a list for its potential control measure and sent to the calculators that assign the relevant costs. This implementation has two calculators available; the assault position calculator and the attack position calculator. Each calculator holds the methods that assign costs based on the planning factors. Each node is assigned a cost for each of the planning factors. Costs are stored in separate dictionaries for planning factor. Dictionaries are used instead of directly assigning costs to nodes so that each node only needs to be scored once while the dictionary can be read repeatedly throughout the process. Each specific cost is then normalized. This is done to convert each cost to the same scale so they can be weighted, compared equally, and summed. Each cost is normalized by dividing its assigned score by highest score for that specific criteria. For example, the path cost for an assault position is normalized by dividing its path cost by the highest path cost in the list of potential assault position nodes. Note that normalizing can only be done after the filtering or sorting and cost calculation is completed. This puts each score on a scale from 0 to 1, representing the percentage of the worst it can possibly be. A position with a score of 1 for a specific planning factor means that position has the highest cost or worst score for that factor out of all the potential positions. Once each score is normalized, they are multiplied by their user-defined weights, and summed for the total score of that position. Figure 28 shows the pseudo code for a total cost calculation for a list of potential assault position nodes. The same method shown in Figure 28 is used to calculate the total cost for a list of potential attack positions except that an attack position calculator is used, and the positions are scored based on the attack position planning factors.

```
getTotalCost(nodes, unit) {
     nodes = list of nodes for a particular tactical control measure
     unit = type of unit (ME or SE)

     //Get all the cost dictionaries associated with the provided list of nodes
     assltPosCalculator = Instantiate an assault position calculator
     assltDistDictionary = assltPosCalculator.getAssaultDistanceCost()
     pathDictionary = assltPosCalculator.getPathCost()
     combinedArmsDictionary = assltPosCalculator.getCombinedArmsCost()
     neighborsDictionary = assltPosCalculator.getNeighborsCost()
     supportingArmsDictionary = assltPosCalculator.getSupportingArmsCost()

     foreach node in nodes
          totalCost = (pathDictionary[node] / maxPathCost) * pathWeight
               + (assltDistDictionary[node] / maxAssltDistCost) * assltWeight
               + (combinedArmsDictionary[node] / maxComArmsCost) * combArmsWeight
               + (neighborsDictionary[node] / maxNeighborsCost) * neighborsWeight

          if(unit == SE) {
               totalCost += (supportingArmsDictionary[node] / maxSptingArmsCost) * sptingArmsWeight
          }
     return dictionary with totalCost for every node from the list
}
```

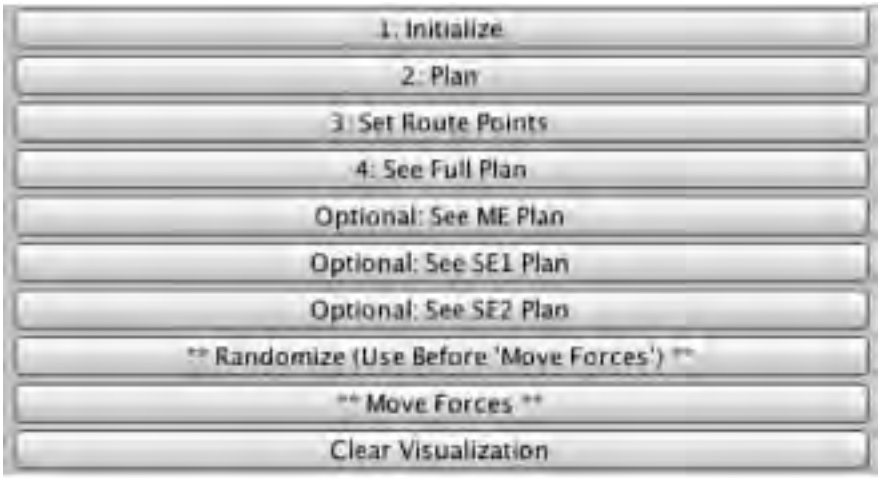
Figure 28. Pseudo Code: Assault Position Total Cost Calculation

Once a set of nodes is scored based on the planning factors for its type of control measure and assigned a total cost, the total cost dictionary is then iterated through to find the lowest scoring node which is selected as the position for that type of control measure. Once all tactical control measures have been selected, the maneuver plan is essentially complete. The only addition to the maneuver plan is the paths between tactical control measures. These paths were already constructed and scored, so no extra work is done. We simply assign the already constructed ABPaths to connect the control measures and complete our plan.

D. USING THE EDITOR TO DEVELOP A MANEUVER PLAN

This section will detail the step-by--step process that the maneuver planner is executing to produce a plan. Figures 29 depicts the button panel that the user will use to initiate the planner. Selecting the “Initialize” button will collect the user inputs and assign

all the relevant planning variables. The “Plan” button initiates the planner which executes each of the steps described below. All button functionalities are described in Appendix A. Note that Figures 32–38 were included to show a behind the scenes look at what each step in the execution is producing. These visualizations will not actually be seen by the user during normal use of the planning tool. The full plan visualization shown in Figure 39 is the actual visual output of the planning tool after selecting the “See Full Plan” button.



The buttons labeled 1 through 4 must be selected in order. The buttons labeled “Optional” give the user extra flexibility to view specific parts of the plan. The last three buttons are used to move the forces around the terrain to explore different scenarios.

Figure 29. Button Editor Used to Develop a Maneuver Plan

The step-by-step process to develop a maneuver plan is as follows:

1. Set user inputs. Navigate to the friendly unit in the Unity3d Hierarchy window. Once selected, the editor window in Figure 30 will appear in the Unity3d Inspector window. The user is required to set the units, user inputs, and both assault and attack position weights. All entries are described in Appendix B. The input weights reflected in the figure below are used for Figures 32–38.

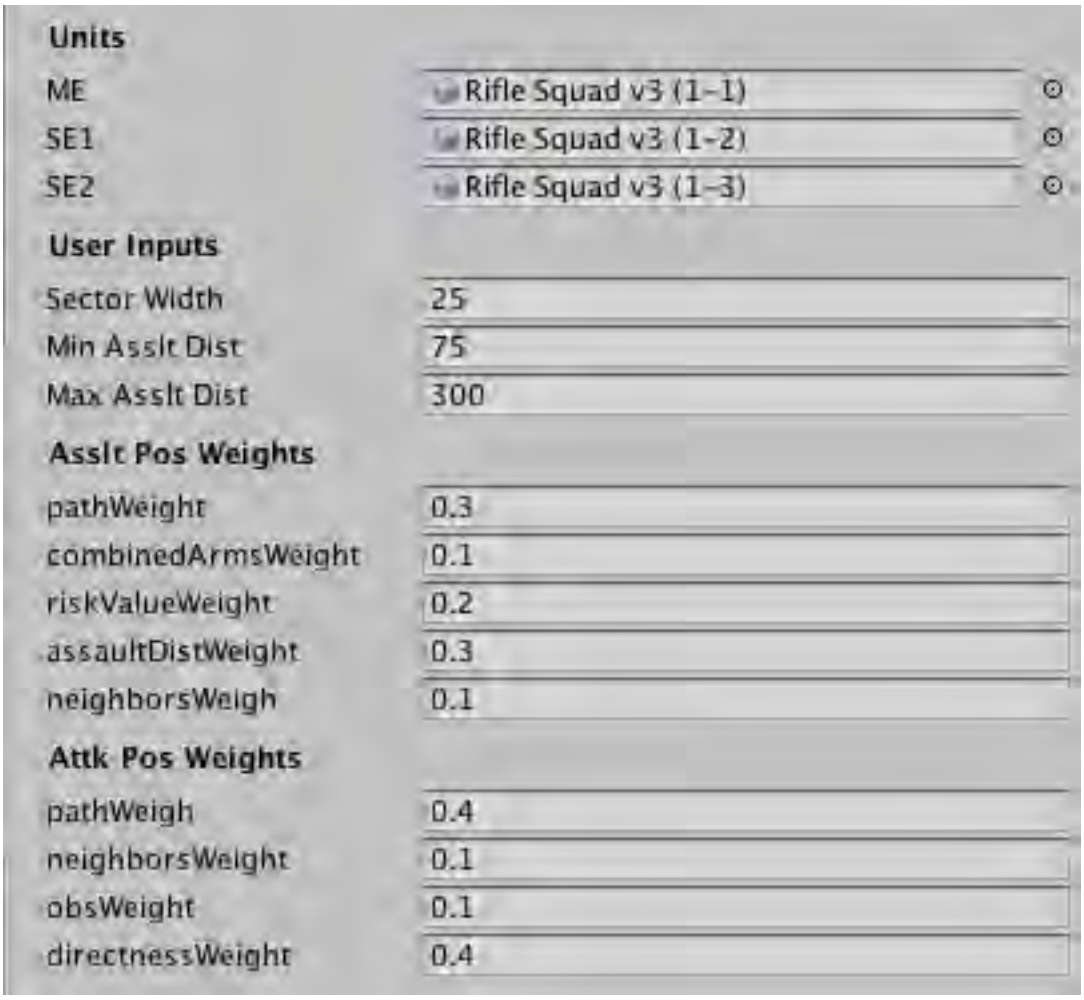
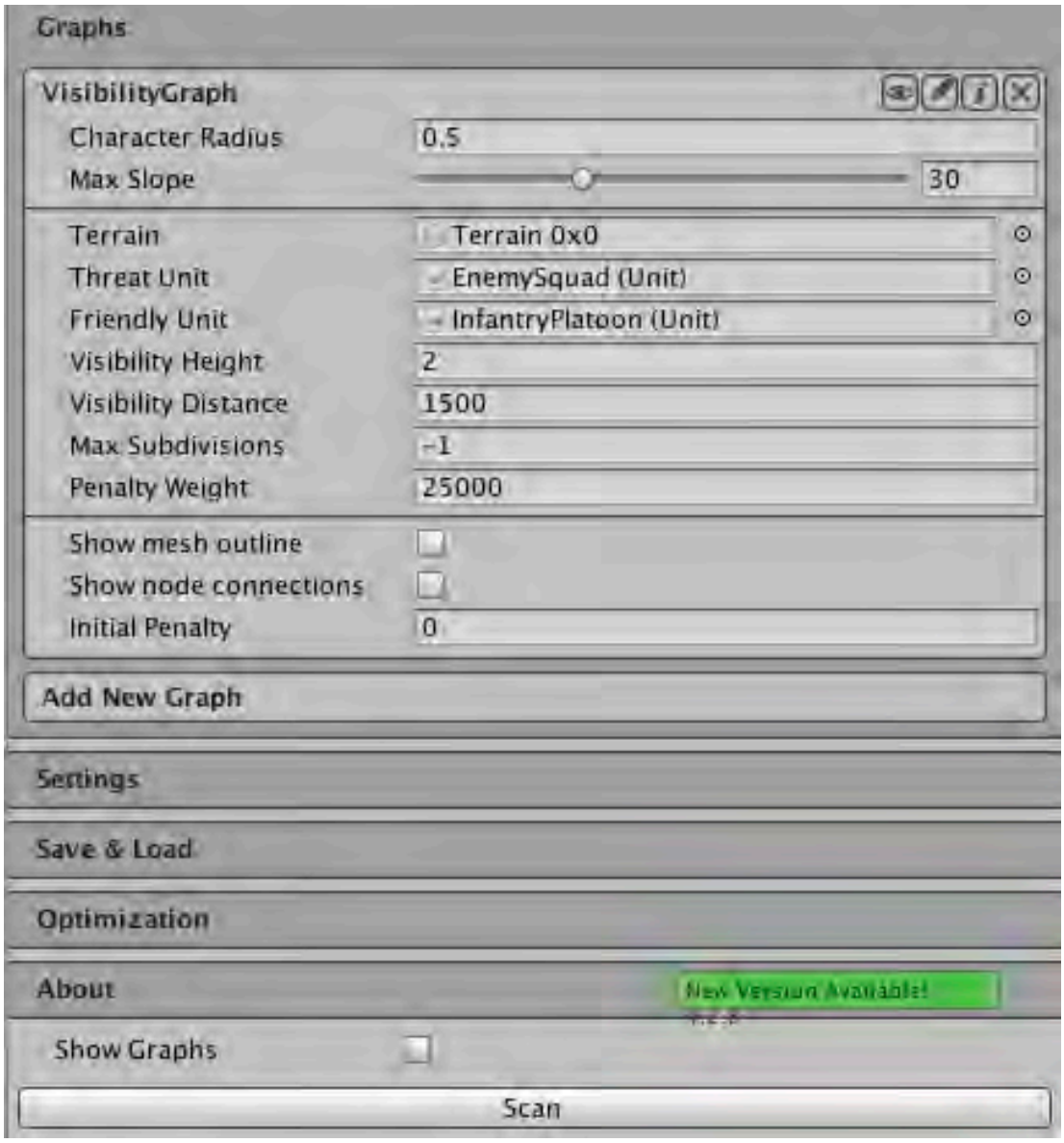


Figure 30. User Input Editor Located in the Unity3d Inspector Window

2. Generate the VisibilityGraph. Navigate to the A* object in the Unity3d Hierarchy window. Once selected, the graph editor in Figure 31 will appear in the Unity3d Inspector window. The user is required to set the terrain, and the threat and friendly units. The user may change the other additional options, or they will simply use the default value. All entries are described in Appendix C. Once set, click the “Scan” button to generate the graph.



This panel is located under the A* object in the Unity3d Hierarchy window.

Figure 31. VisibilityGraph Editor

3. Filter and sort potential assault positions into sectors (See Figure 32).

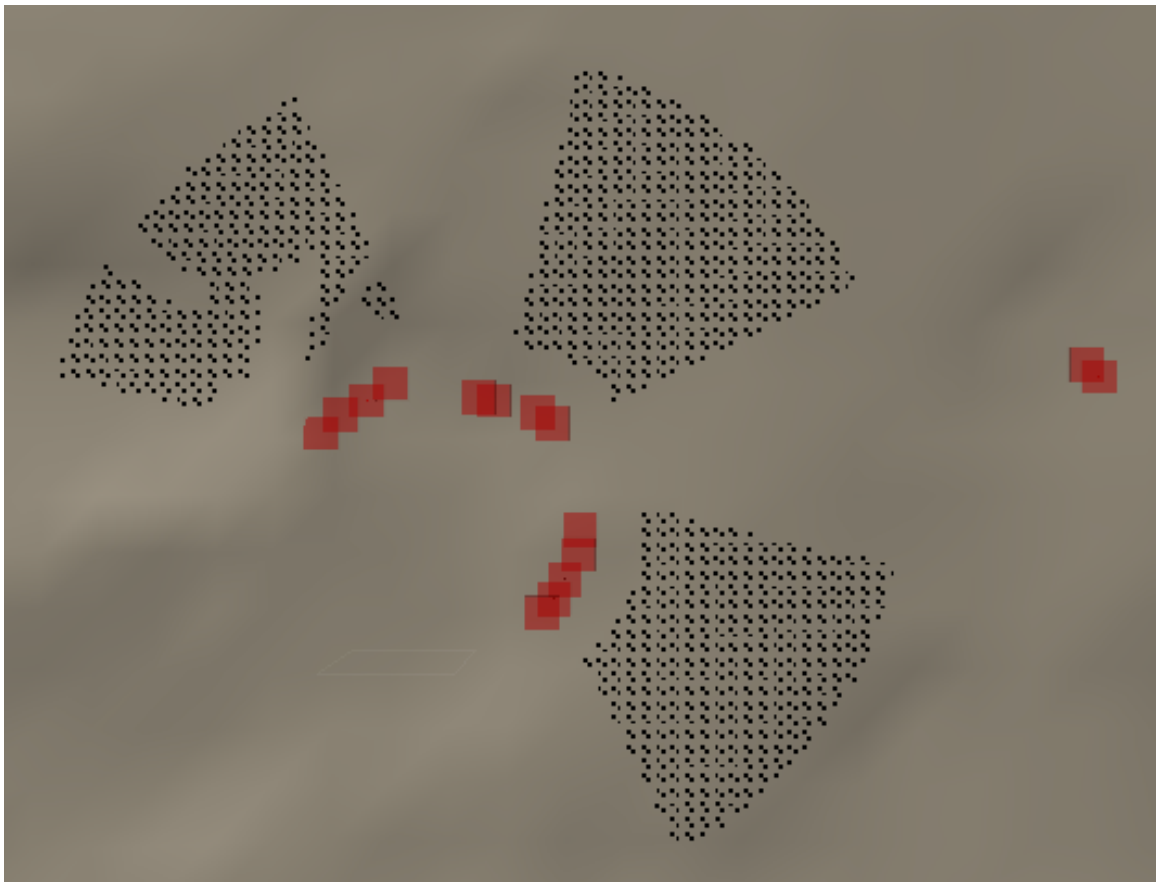
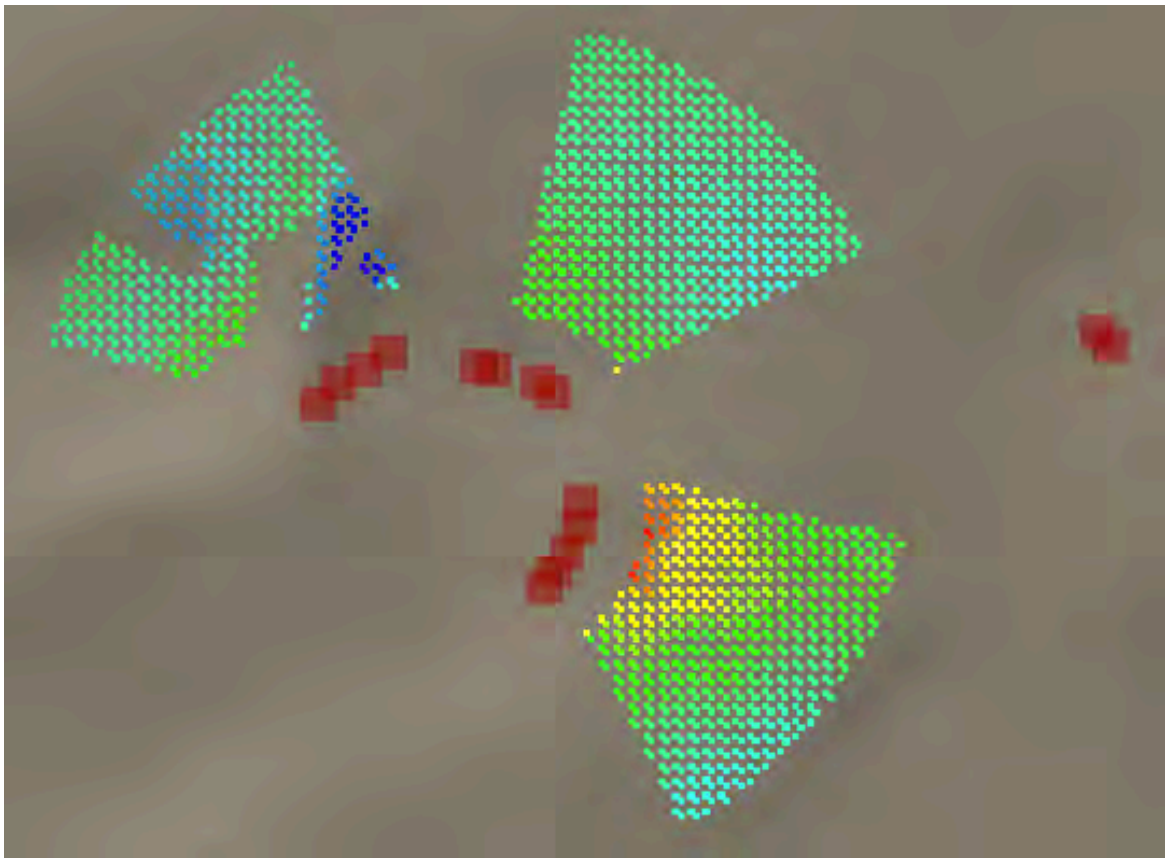


Figure 32. Visualization of Filtered, Sorted, and Unscored Potential Assault Position

4. Score all potential assault positions based on the assault position planning factors (See Figure 33).



The color scheme works from blue to red where blue is the lowest scoring or most desirable positions, and red is the highest scoring, or least desirable positions.

Figure 33. Visualization of the Assault Position Sectors After Being Scored Based on the Assault Position Planning Factors

5. Select the lowest scoring position (most desirable) as the ME assault position (See Figure 34).

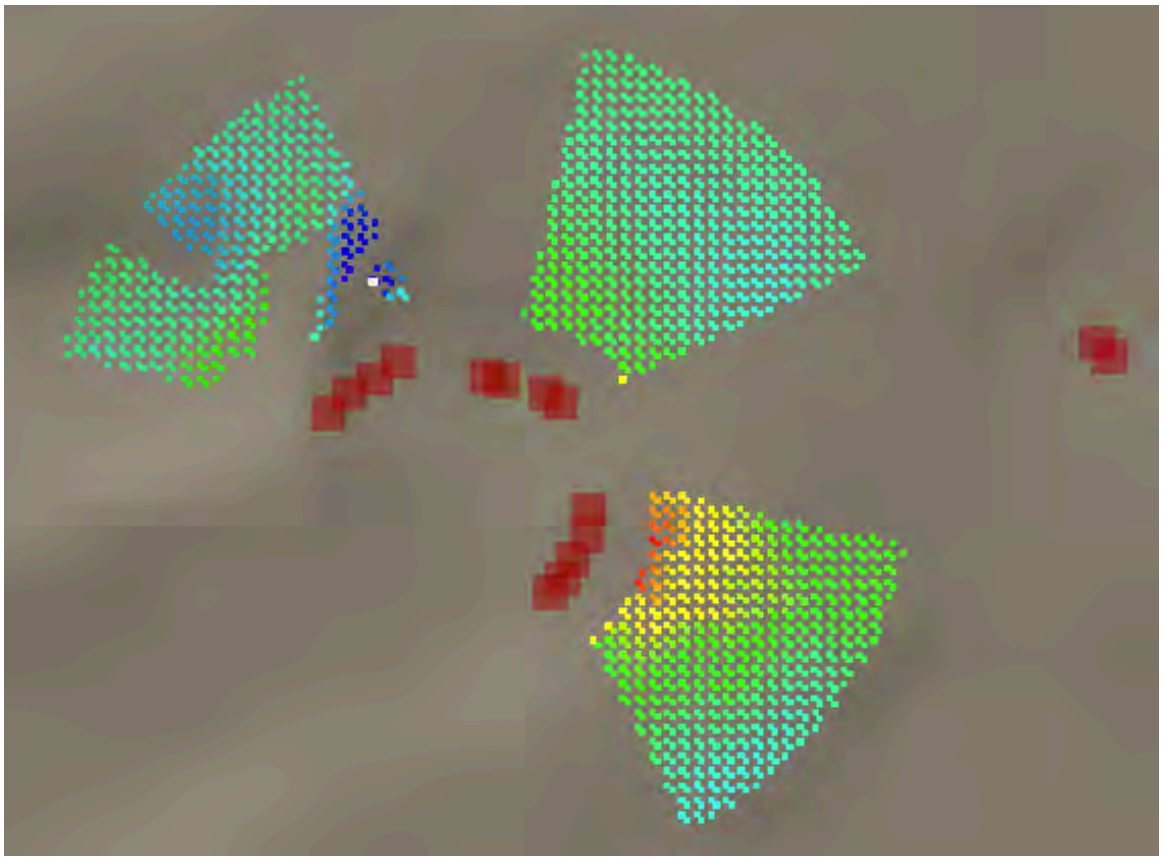
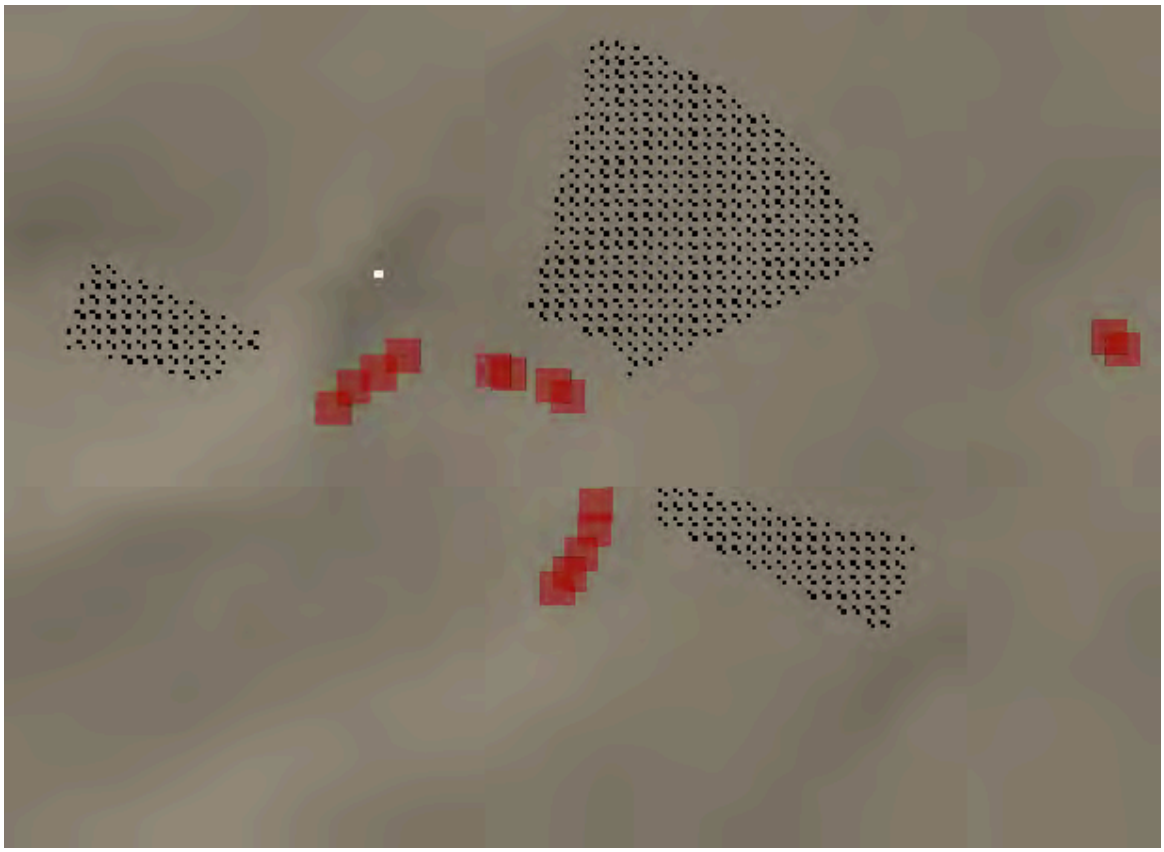


Figure 34. Visualization of the Selected ME Assault Position in White

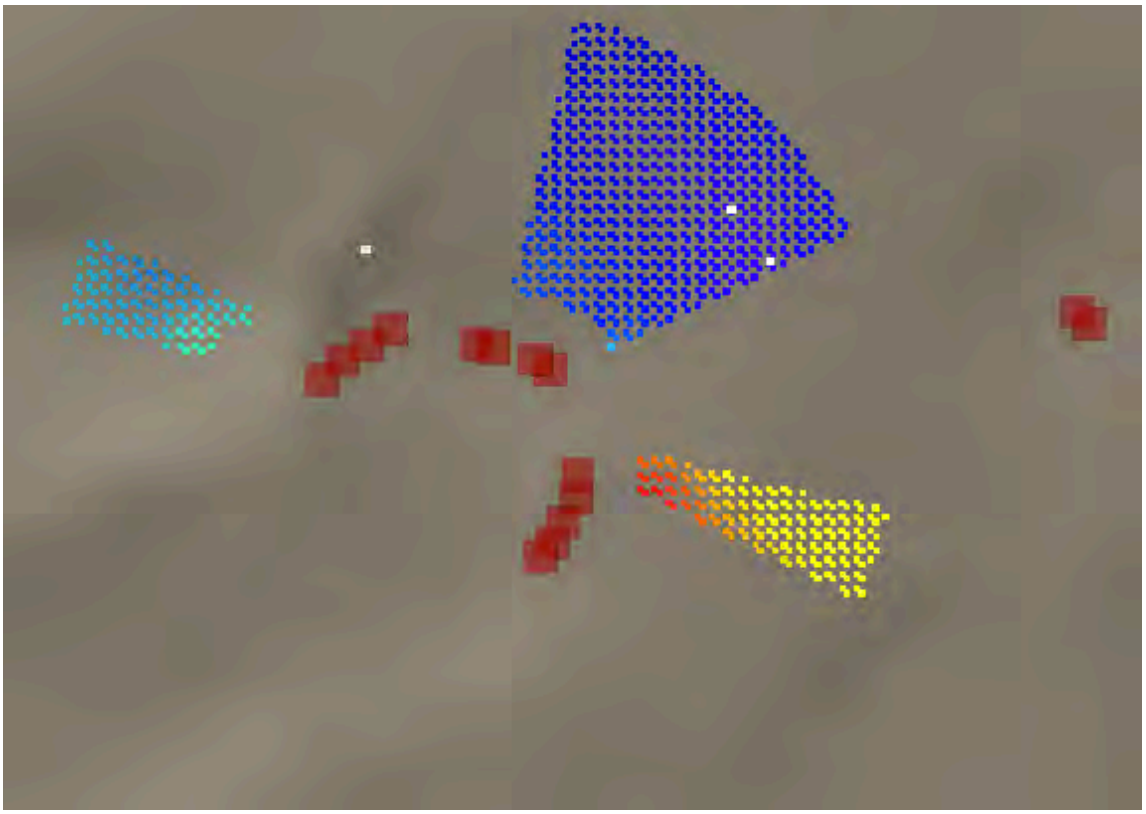
6. Filter out remaining assault positions that are at risk of friendly fire from the selected ME assault position (See Figure 35).



The white square is a reminder of where the ME assault position is located.

Figure 35. Assault Position Sectors After Nodes Have Been Filtered for Being at Risk of Friendly Fire

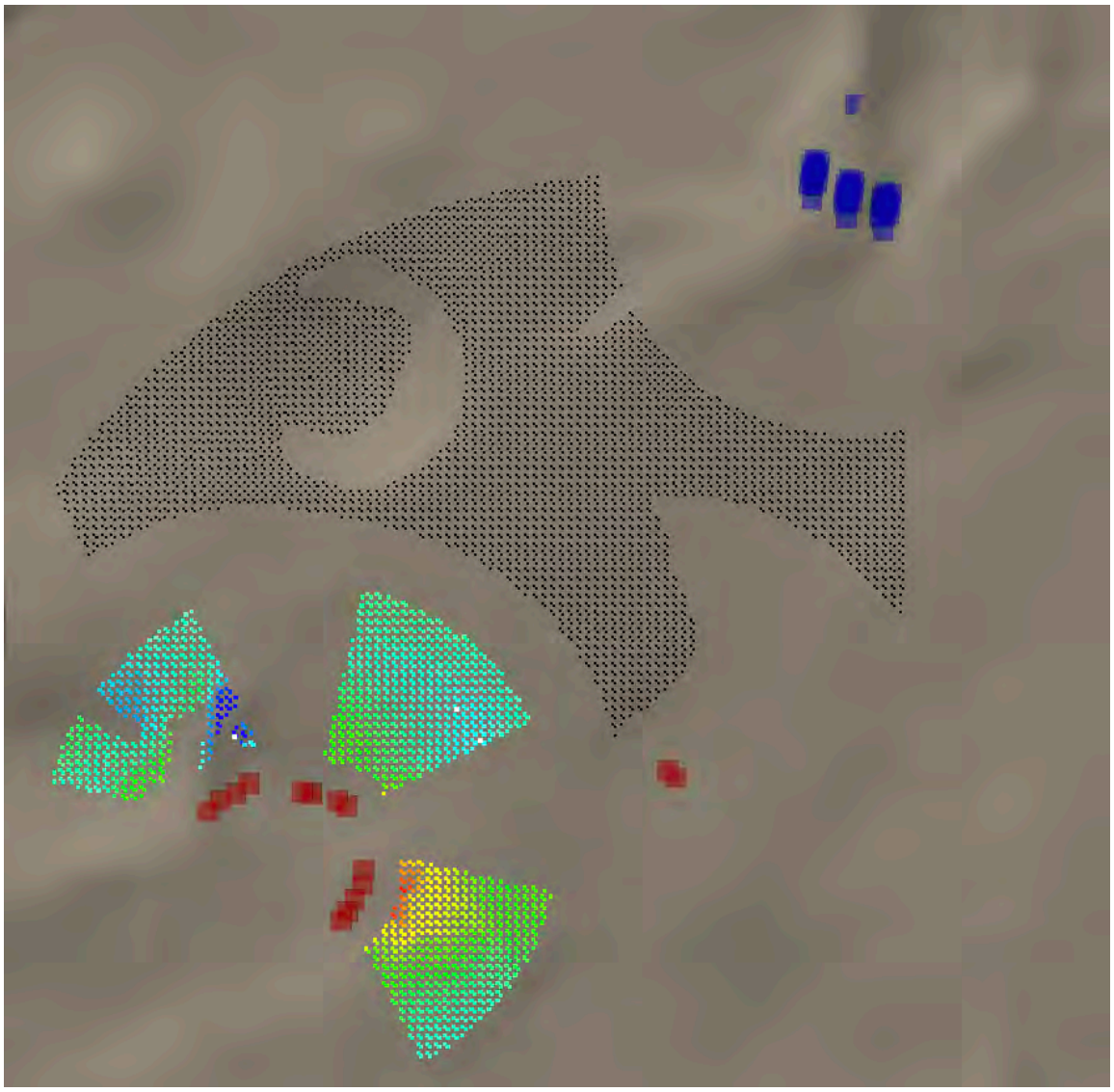
7. Score remaining assault position nodes with the addition of the supporting arms cost and select the two lowest scoring nodes from the remaining assault position nodes as the SE1 and SE2 assault positions (See Figure 36).



Note that the additional supporting fires cost is assigned to these positions. Positions closer to a 90-degree offset are more desirable as identified by the blue positions. The two white squares in the center sector represent are the two positions selected for the two supporting effort units.

Figure 36. Assault Position Sectors After Being Scored for the Remaining Two Supporting Effort Units

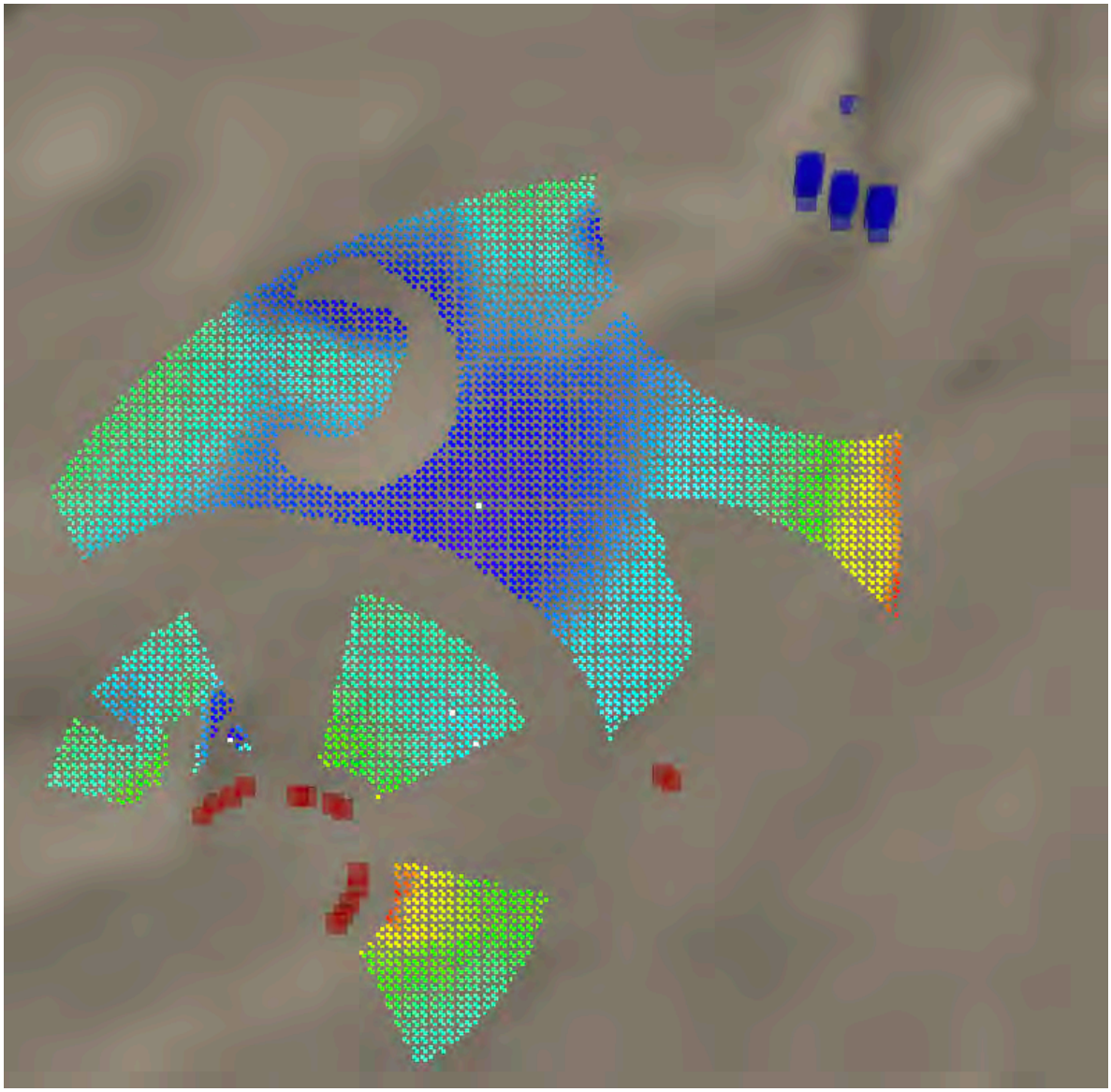
8. Filter out nodes that do not meet minimum threshold criteria for an attack position (See Figure 37).



The white squared represent the previously selected ME assault position and supporting effort assault positions.

Figure 37. Filtered and Unscored Positions in Black That Have Met the Minimum Criteria to be Considered for an Attack Position

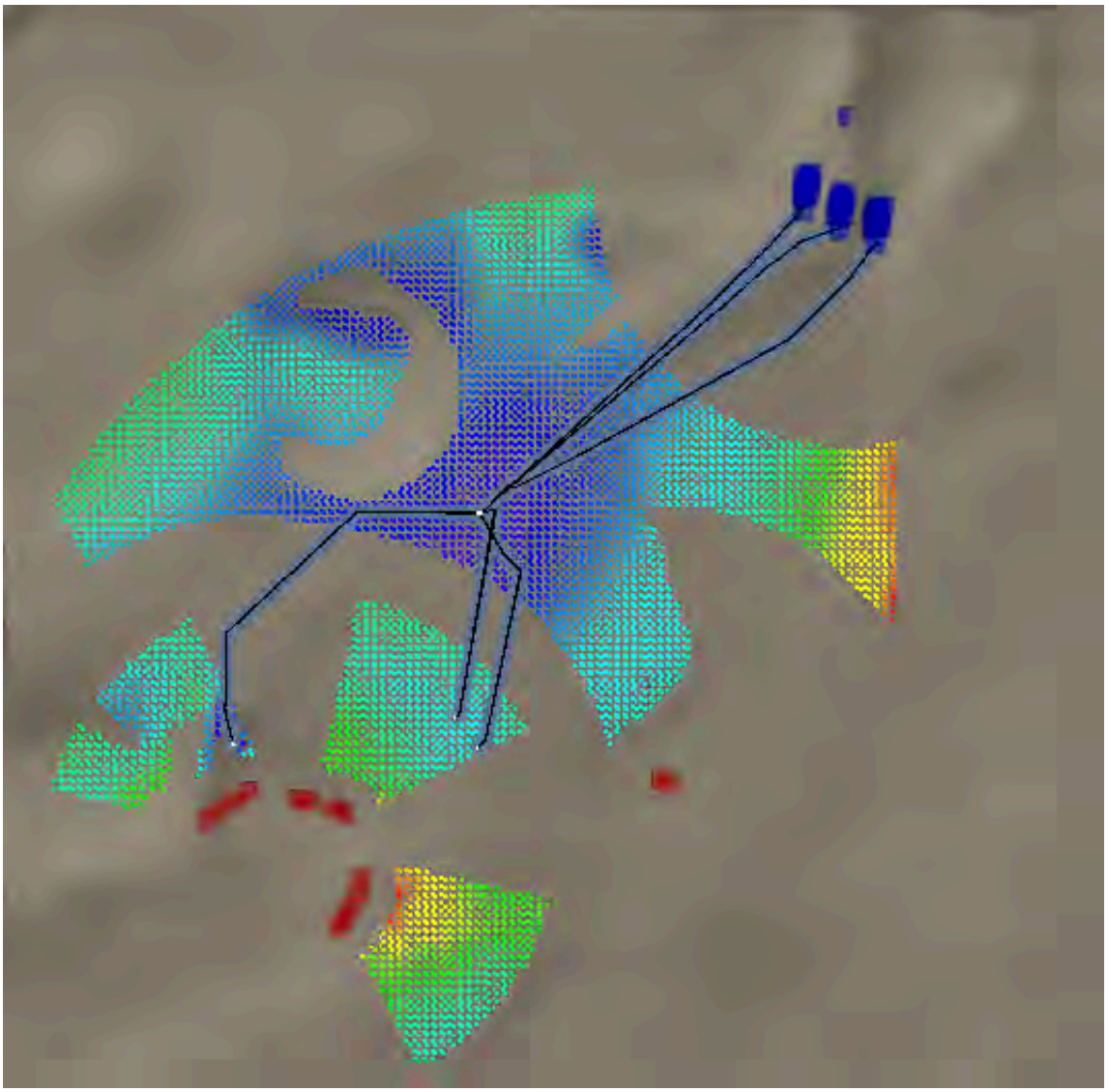
9. Score all potential attack positions based on the planning factors and select the lowest scoring node as the platoon attack position (See Figure 38).



The white square represents the position selected as the platoon assault position. The color scheme works from blue to red where blue is the lowest scoring positions, and red is the highest scoring or least desirable positions.

Figure 38. Visualization of the Potential Attack Positions After Being Scored

10. Add routes and view the full plan (See Figure 39).



The type of attack is a right flank from the perspective of the friendly unit.

Figure 39. Visualization of the Full Maneuver Plan.

E. EXAMPLE RESULTS

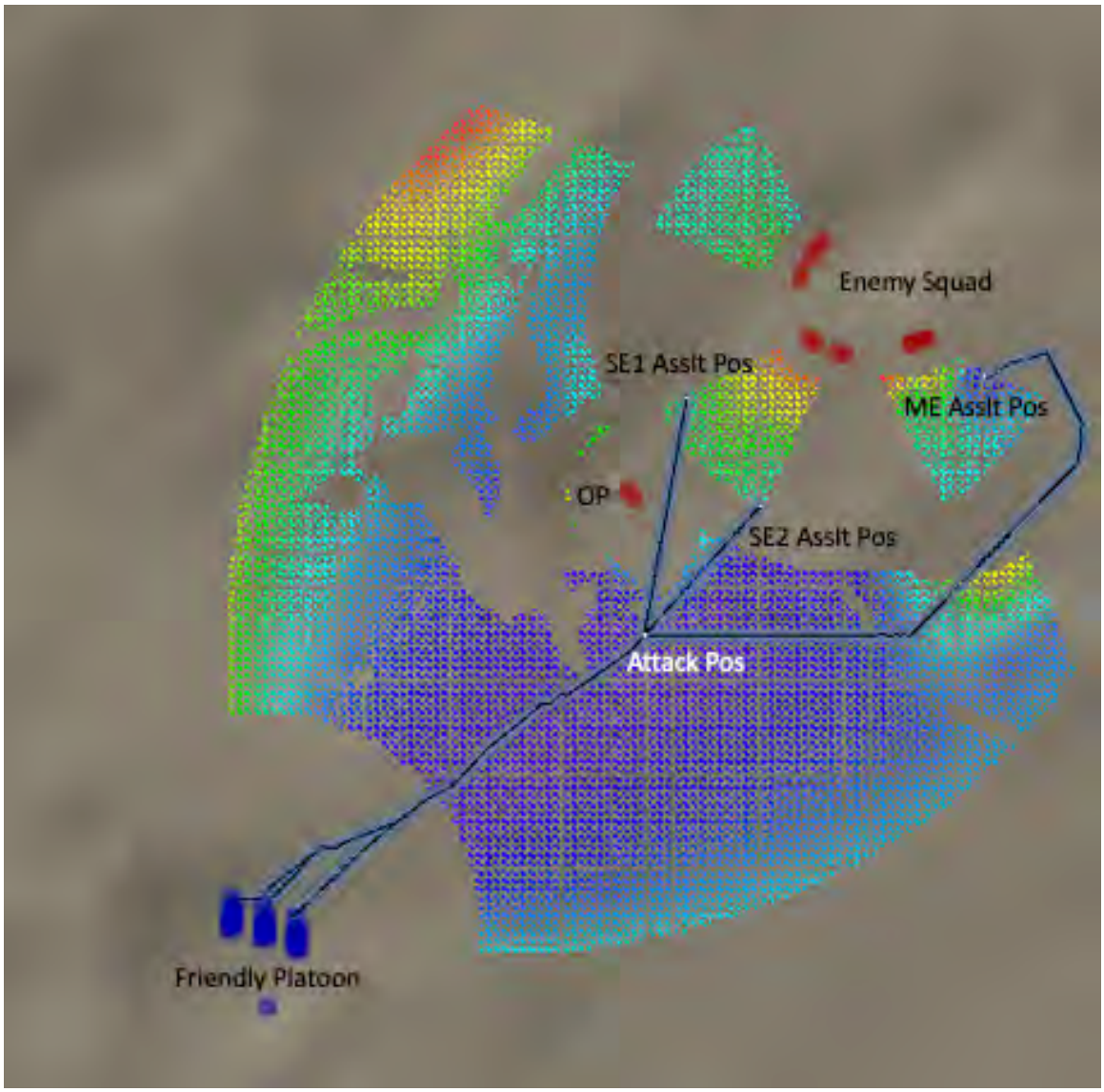
The following examples depict maneuver plans that were developed for different terrains and different enemy and friendly locations. The weights used are found in the tables preceding each figure. It should be noted that the choice of what type of attack to use (frontal, flanking left, flanking right) is made by the planner itself based on the threshold criteria, planning factors, and the terrain constraints, without any explicit direction or input from the user.

1. Permissive Environment

Figure 40 depicts a basic scenario with permissive terrain for the friendly unit identified by the large blue area surrounding the attack position. The type of attack is a right flanking attack with both SE units supporting from the frontal sector. The attack position and the route to the attack position for the entire friendly platoon crosses directly through the middle of the permissive area due to the short distance and low risk. Table 1 shows the user input weights used in this scenario.

Table 1. User Inputted Weights for Figure 40

Assault Position Weights	
Path Weight	0.1
Combined Arms Weight	0.1
Risk Value Weight	0.3
Assault Distance Weight	0.3
Neighbors Weight	0.2
Attack Position Weights	
Path Weight	0.2
Neighbors Weight	0.3
Directness Weight	0.4
Observer Weight	0.1



The large blue area represents a permissive area resulting in the attack position being located in the permissive area as well as the route to the attack position crossing directly through the center of the permissive area.

Figure 40. Scenario 1: Permissive Environment

2. Frontal Attack

Figure 41 depicts a basic frontal attack. This terrain has a much smaller permissive area than Figure 40 located at the bottom of the figure where the attack position is located. Notice that the assault position sector on the right of the enemy position is higher scoring

(colors closer to red) than the other two sectors and is completely avoided by any assault positions. Table 2 shows the user inputted weights in this scenario.

Table 2. User Inputted Weights for Figure 41

Assault Position Weights	
Path Weight	0.2
Combined Arms Weight	0.1
Risk Value Weight	0.2
Assault Distance Weight	0.3
Neighbors Weight	0.2
Attack Position Weights	
Path Weight	0.3
Neighbors Weight	0.1
Directness Weight	0.4
Observer Weight	0.2

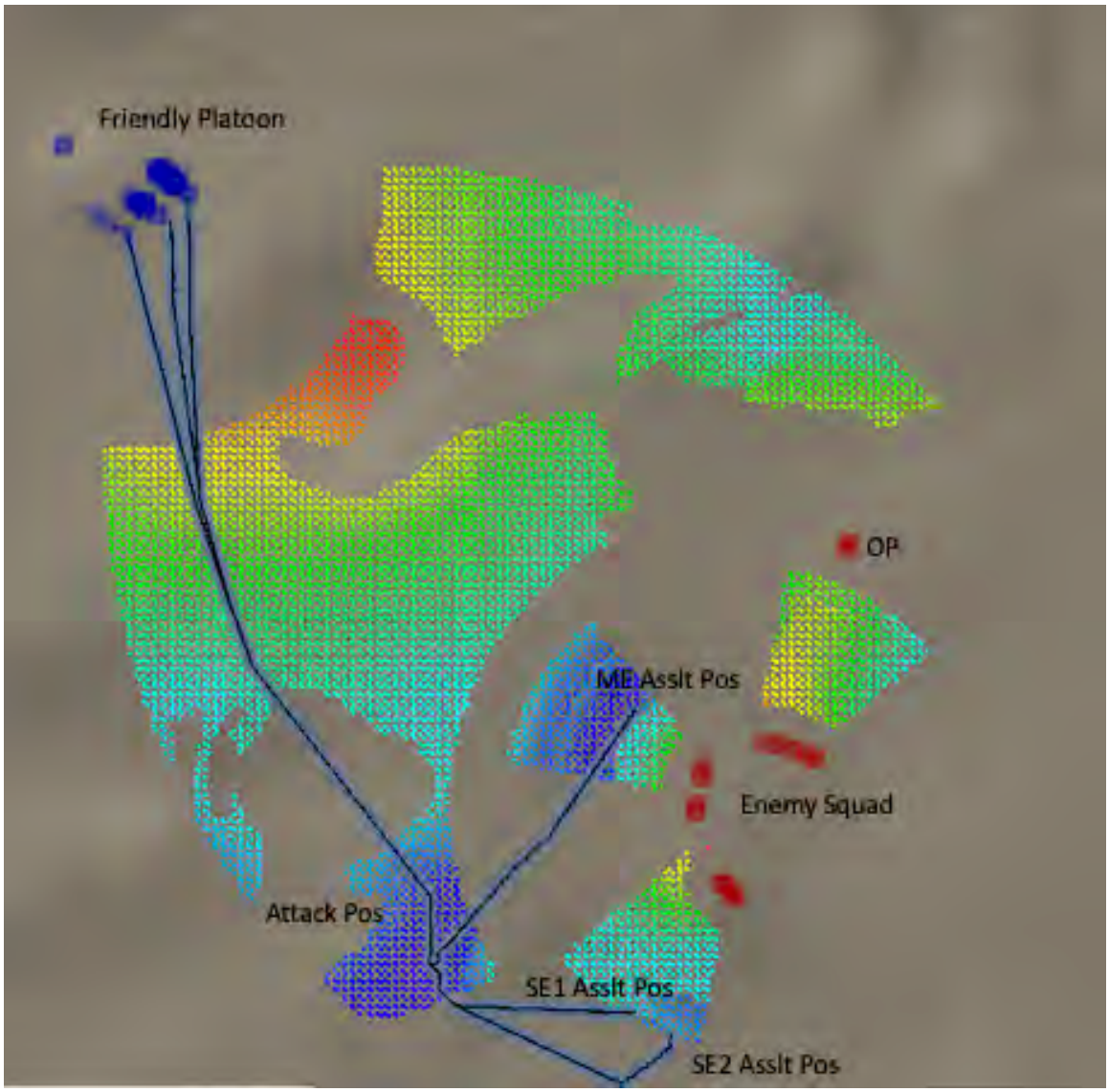


Figure 41. Scenario 2: Frontal Attack

3. Right Flanking Attack

Figure 42 depicts a basic right flanking attack. Similar to permissive environment scenario in Figure 40 and Figure 41, this scenario also has an attack position and route that are in-line with the enemy position. Notice that the assault position sector on the right of the enemy position and the right side of the attack position nodes are higher scoring (colors closer to red), indicating that terrain towards the right side of the map is potentially less desirable for the friendly unit. Table 3 shows the user inputted weights used in this scenario.

Table 3. User Inputted Weights for Figure 42

Assault Position Weights	
Path Weight	0.3
Combined Arms Weight	0.1
Risk Value Weight	0.2
Assault Distance Weight	0.3
Neighbors Weight	0.1
Attack Position Weights	
Path Weight	0.4
Neighbors Weight	0.1
Directness Weight	0.4
Observer Weight	0.1

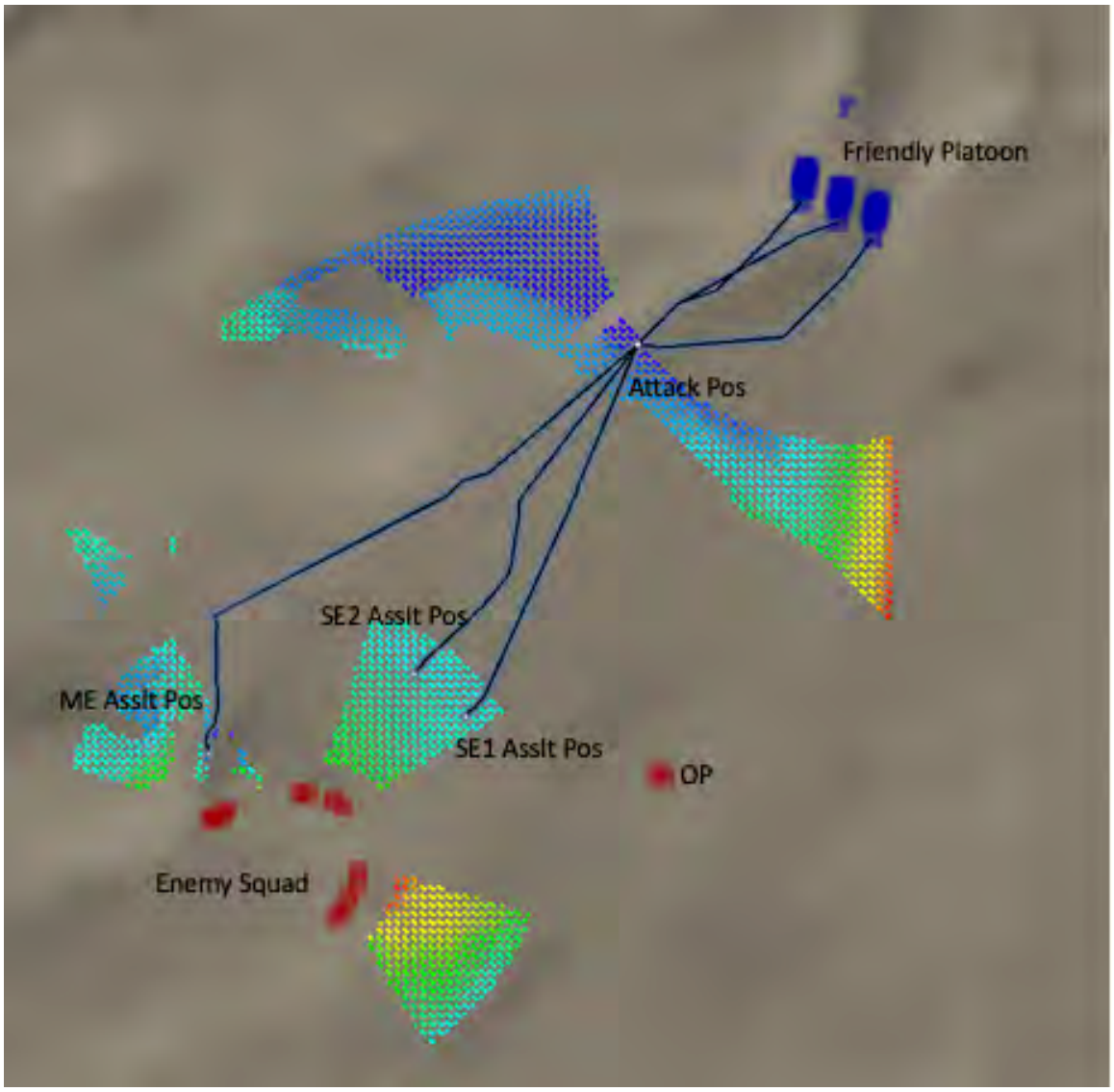


Figure 42. Scenario 3: Basic Scenario Depicting a Right Flanking Attack

4. Risk versus Distance Tradeoff

Figure 43 depicts a right flanking attack with the SE units supporting from the frontal sector and an attack position located in a large permissive area similar to Figure 40 and Figure 42. Notice that the routes from the attack position to the SE1 and SE2 assault positions takes a wide approach avoiding the area that can presumably be observed or effected by the enemy OP and enemy position. This demonstrated a common tradeoff between risk and distance. In order to reduce risk, a route will likely get longer. Similarly,

reducing distance will likely increase risk along the path. Table 4 shows the user inputted weights for this scenario.

Table 4. User Inputted Weights for Figure 43

Assault Position Weights	
Path Weight	0.1
Combined Arms Weight	0.1
Risk Value Weight	0.3
Assault Distance Weight	0.3
Neighbors Weight	0.2
Attack Position Weights	
Path Weight	0.2
Neighbors Weight	0.3
Directness Weight	0.4
Observer Weight	0.1

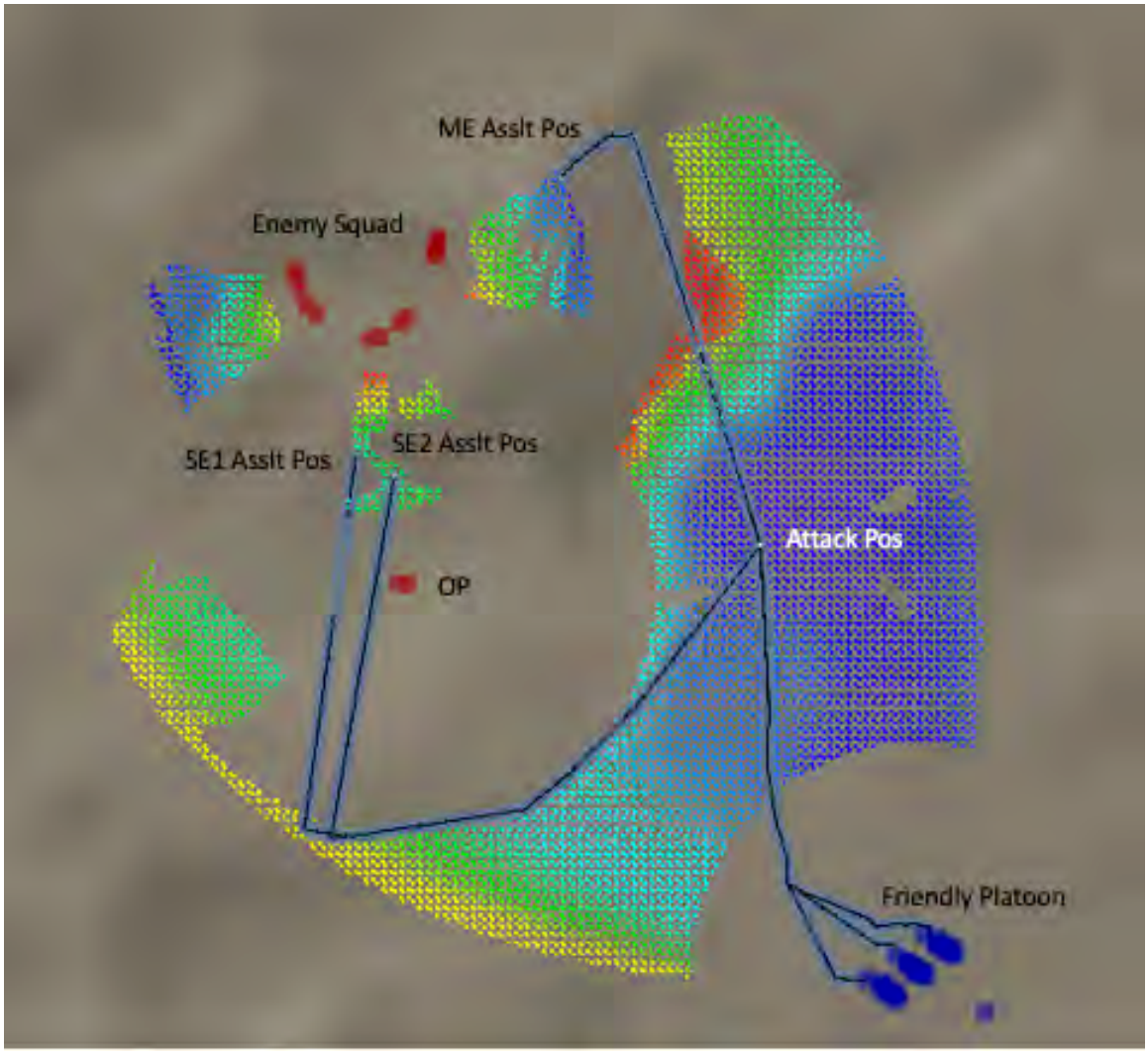


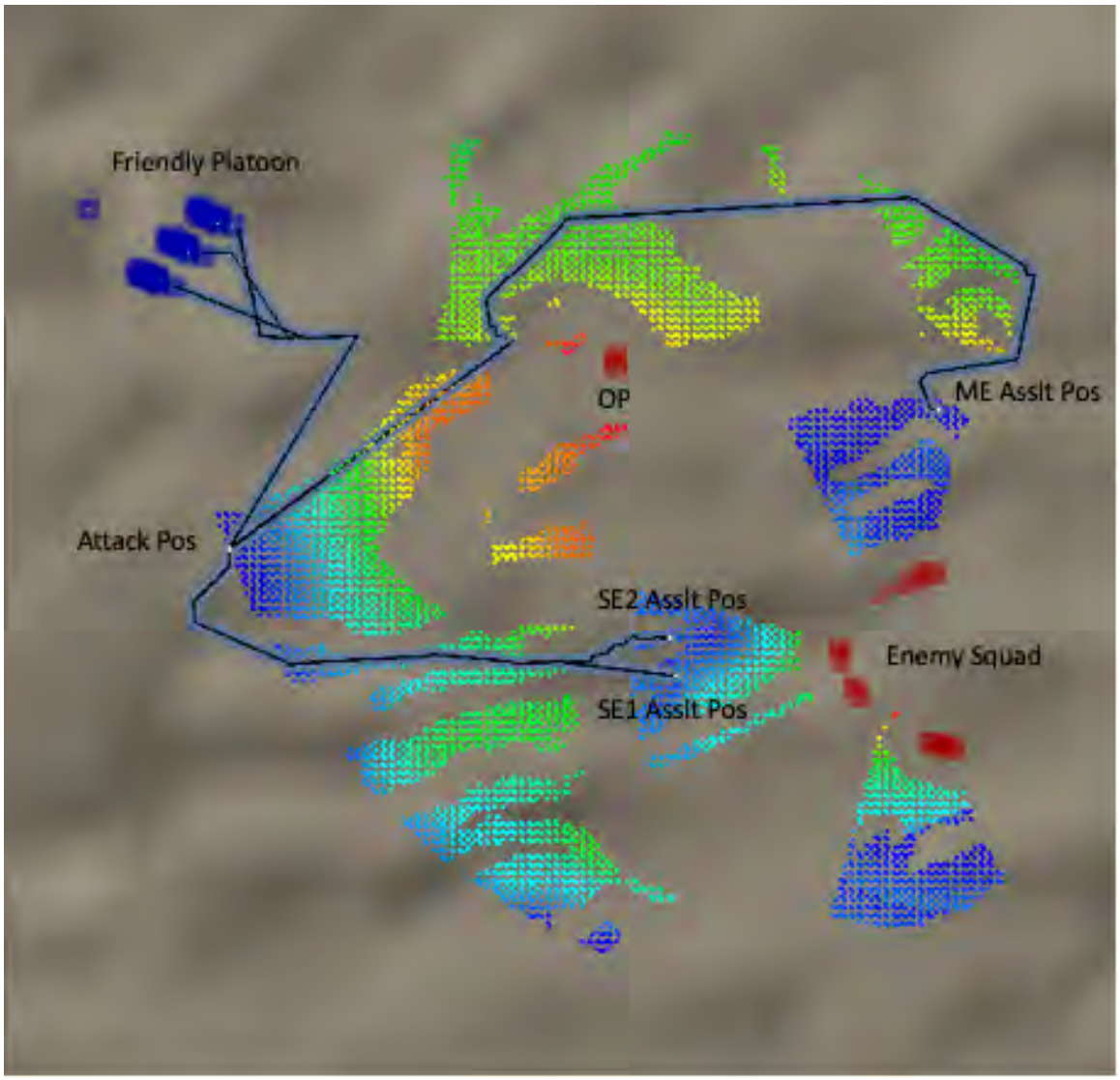
Figure 43. Scenario 4: Risk Versus Distance Tradeoff

5. Left Flanking Attack in Mountainous Terrain

Figure 44 depicts a left flanking attack in mountainous terrain. The mountainous terrain consists of ridges that flow from the left to right of Figure 44. This creates regions between and surrounding each of the ridges that can have very different costs based on small differences in their relation to the enemy position and OP. Additionally, the terrain has multiple pockets of unwalkable nodes that limit agent mobility. Both of these considerations result in longer paths that appear to take a roundabout approach like the path from the attack position to the ME assault position. Table 5 shows the user inputted weights for this scenario.

Table 5. User Inputted Weights for Figure 44

Assault Position Weights	
Path Weight	0.4
Combined Arms Weight	0.1
Risk Value Weight	0.2
Assault Distance Weight	0.2
Neighbors Weight	0.1
Attack Position Weights	
Path Weight	0.1
Neighbors Weight	0.2
Directness Weight	0.5
Observer Weight	0.2



Notice that the most permissive paths seem to take a round-about route to the assault positions.

Figure 44. Scenario 5: Mountainous Terrain

VI. RECOMMENDATIONS AND CONCLUSIONS

COMBATXXI has two shortcomings that were addressed in this thesis. First, entity maneuver is scripted by scenario developers which prolongs the development process and reduces the time available to do simulation runs for analysis. Second, COMBATXXI lacks a robust prototyping capability that can be used to construct and test new behaviors. This thesis engaged both of these shortcomings by first using a surrogate environment to develop a maneuver planner, then attempting to transfer the relevant control measures and route data from the surrogate environment to COMBATXXI. Transferring data from external environments to COMBATXXI is not an easy process. Building upon this capability is potential future work.

A. RECOMMENDATIONS

1. Choosing a Surrogate Environment

Rapid prototyping will continue to be an area of interest for COMBATXXI developers. Efficient import and export of data between an external environment and COMBATXXI is more difficult than simply organizing information into the proper format and importing without issue. Proper transfer of data requires the user have a deep understanding of both environments and their limitations. This thesis used Unity3d as a surrogate environment because of resident knowledge of the engine. However, the question of what environment provides the best external environment for rapid prototyping is still to be answered. With this in mind, any external environment considered for this task should have some basic capabilities. First, it should have a robust and easily understood visual debugging capability. This cannot be overstated. The ability to receive errors and quickly track down and fix is essential for rapid prototyping. Second, it should have a strong community of users and formal documentation to support development. Last, it should be easily attainable and maintainable for the DoD. Complicated licenses and user agreements will likely add barriers and ultimately slow down a process that is meant to be quick. While this is less of a problem for the commercial modeling and simulation community, it will

likely be an issue for the DoD that will need to be addressed at the policy level to improve agile development.

2. Computational Efficiency

A significant portion of tactical position selection is large spatial queries. In this thesis, terrain size was limited in size to approximately 2.5 x 2.5 km. This can result in a terrain representation that consists of approximately 8000 to upwards of 30,000 nodes. Iterating, filtering, sorting, and scoring this number of nodes can get burdensome. Limiting the number of nodes is logical, but this comes at a cost of reducing the fidelity at which we can reason about the terrain. Reducing the number of nodes means that each node now represents a larger area. If the area is too large, the annotated information can become inaccurate. This implementation is limited by its efficiency. Computations can take up to three minutes to compute a maneuver plan. It is recommended that any exploration into tactical position selection or multi-unit maneuver include a plan to reduce overhead and optimize computational efficiency. This may naturally include the use of common or custom data structures and a framework for data flow and manipulation. The use of parallel processes should also be investigated since many of the algorithms used in this thesis look to be amenable to parallel implementation.

B. FUTURE WORK

Because this thesis touches a number of areas with respect to the COMBATXXI simulation, future work can lead in many different directions. This section will recommend three areas of research that would benefit the modelling and simulation community and the cadre of COMBATXXI developers.

1. Transferability

Transferability of data from an external environment into COMBATXXI is not a trivial process. Different data types require different types of manipulation to properly pack and transfer. A user will likely need to execute multiple processes, data conversions, and format organization to fully transfer a behavior with its associated environmental and scenario data. Additionally, a developer must be well versed in both environments to

accomplish the task efficiently without interrupting development workflow. There is a need for a deep exploration into the full spectrum of available surrogate environments that can be used for prototyping. Included in this exploration should be an analysis of the capabilities and limitations of each environment with respect to the capabilities and limitations of COMBATXXI as well as the detailed processes that would be required to fully transfer behaviors from one environment to the other.

2. Experimentation

This tactical position selection system in conjunction with the WOMBATXXI planners has the ability to produce control measures, develop mission plans from those control measures, and modify the mission plan using the fire support enhancement planner to create a doctrinal or plausible maneuver plan. Weighting planning factors results in different plans and logically, different operation outcomes. WOMBATXXI provides the framework to conduct an analysis of alternatives for different maneuver plans which emerged from changing the planning weights. A measure such as casualties or time to complete the mission are reasonable performance measure to evaluate a given maneuver plan. A Monte Carlo methodology could be used to evaluate a maneuver plan over multiple runs where the random sampling within the simulation would be the weighted values for the planning factors. The statistical output would identify the maneuver plan (series of tactical positions and weights) that produced the best plan based on the specified performance measure. A similar method could be used to identify plans that maximized enemy casualties or any other desirable metrics.

3. System Extensions

Both WOMBATXXI and the tactical position selection system provide areas to expand the capability of the system. Relevant extensions to WOMBATXXI in particular can be found in LtCol Byron Harder's dissertation [2].

The tactical position selection system in this implementation is scoped to support a platoon size deliberate attack on a static defensive position. Maneuver planning is not isolated to platoon operations and the system would benefit from expanded capability to reason about larger sized units in different types of offensive and defensive operations.

Additionally, different size units, units with different capabilities, and different types of operations all require tailored planning factors that will allow the tactical position selection system to properly reason about the situation. WOMBATXXI provides the necessary framework to reason about larger units and has a task dictionary that can support any number of operational tasks [2].

4. Emergent Behaviors

Due to the limited input data, intentional ambiguity in the assignment of assault position sectors, user defined weights, and natural uncertainty that comes with analyzing terrain, the planning system output has emergent behaviors, specifically in the type of attack. Such emergent phenomena are described by Ilachinski as simple behaviors that emerge from complex situations either through a "self-organized ecology" of factors within the simulation or fine-tuning by scenario developers [19]. In this case, a simple maneuver plan that is reflective of reality emerges from the interaction between multiple factors within the system. A user cannot understand the resulting maneuver plan without understanding the system components in play, as well as their relationship with each other. It is in this way that the emergence of different maneuver plans is based on the overall planning structure and not any individual component. A deeper understanding of the effect of each component on the emergence of different behaviors as well as any limitations of the system in this regard would benefit the system. Additionally, the system would benefit from an exploration of any thresholds that exists within the input data or user defined weights that have an effect on the resulting maneuver plan.

C. CONCLUSION

Although control measures and route data were not successfully transferred between Unity3d and COMBATXXI, a working prototype which fits within the WOMBATXXI framework that produced a platoon maneuver plan against a static defensive position was developed. It is difficult to formally validate the output of this type of system because a truly good maneuver plan is somewhat subjective. Given a tactical situation and mission, individual subject matter experts will likely develop different plans to accomplish the mission. However, we believe the goal of constructing simple, plausible,

and doctrinal maneuver plans was accomplished. While this prototype explored one method of developing maneuver plans, there is still much more that can be accomplished in this area including the exploration of data transfer between simulation environments, experimentation, and extensions to this system and WOMBATXXI.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. BUTTON PANEL

This appendix details the functionality of each button in Figure 45 below. This button panel is used by the user to organize input data, assign relevant variable values, and initiate the planning system. The buttons numbered 1 through 4 must be selected in order to develop a complete maneuver plan. The three "Optional" buttons allow the user to visualize the maneuver plan for specific subunits within the maneuver plan. The "Randomize" and "Move Forces" buttons allow the user to easily move friendly and enemy forces around the terrain to test the maneuver planner in different terrains and tactical scenarios.

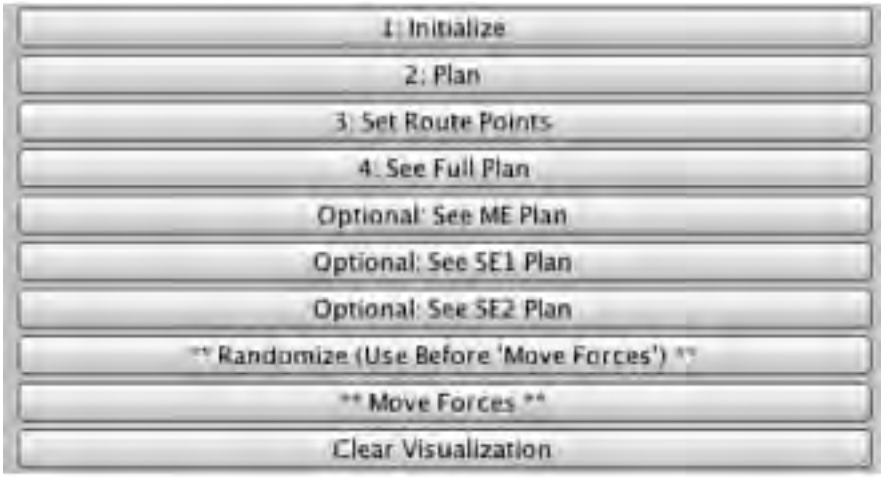


Figure 45. User Button Panel

- Initialize: Gathers all input data from the user input editor. Stores annotated information from the VisibilityGraph including risk and observation values.
- Plan: Initiates the planning system and conducts all filtering, sorting, and scoring described in steps 3–10 in Chapter V, Section D. A complete plan can only be produced if the user input data is set, the VisibilityGraph is scanned, and the "Initialize" button is selected prior to the "Plan" button.

- Set Route Points: Updates the location of the tactical control measure gameobjects found in the Unity3d Hierarchy window based on the completed plan. These control measure gameobjects are picked up by the WOMBATXXI mission planner during its construction of a mission plan.
- See Full Plan: Visualizes the full maneuver plan with an assigned attack position, assault position for each unit, and routes.
- Optional: See ME Plan: Visualizes only the ME plan including its attack position, assault position, objective, and routes.
- Optional: See SE1 Plan: Visualizes only the SE1 plan including its attack position, assault position, objective, and routes.
- Optional: See SE2 Plan: Visualizes only the SE2 plan including its attack position, assault position, objective, and routes.
- Randomize (Use Before 'Move Forces'): Changes the seed value of the random number generator which is used to select random positions on the terrain to set as the friendly and enemy locations.
- Move Forces: Physically moves the friendly and enemy unit gameobjects to the new (random) positions on the terrain.
- Clear Visualization: Clears all the routes are colored markers on the terrain. This button should be selected to reset a scenario.

APPENDIX B. USER INPUT EDITOR DETAILS

This appendix provides additional details on each user input found in the user input editor depicted in Figure 46 below. It is meant to aid in any extensions to the planning system or user input data. Note that the sum of weights for the assault attack position weights' section must equal 1. See Chapter V, Section C for additional weighting details.

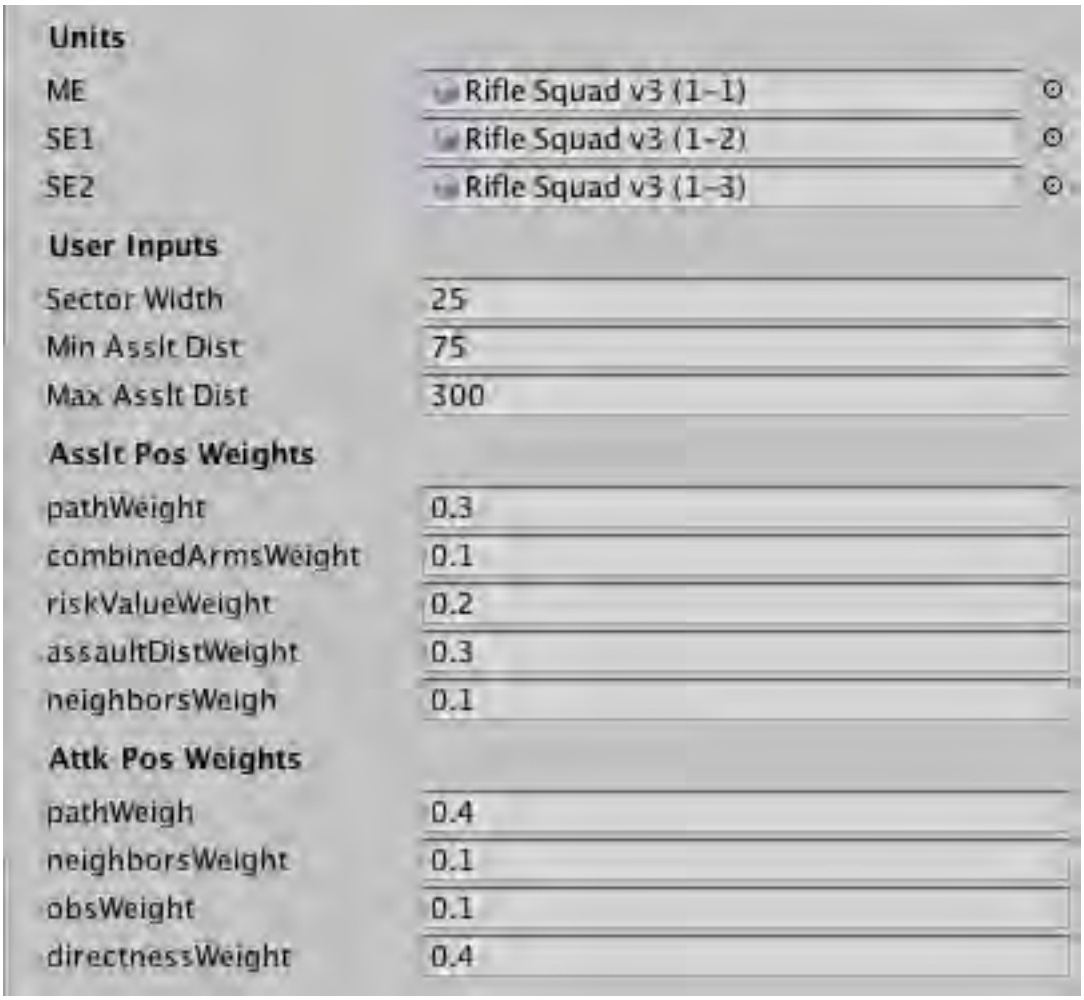


Figure 46. User Input Editor

A. UNITS

- ME: From the Unity3d Hierarchy window, select and drag a squad sized unit into the ME portion of the panel. This unit will be assigned the ME assault position and will be the primary assaulting unit for the maneuver plan. In this case, all squad sized units have identical capabilities, an extension of this system would vary unit capabilities, and the ME unit should have added capability that supports its task to assault the objective.

- SE1: From the Unity3d Hierarchy window, select and drag a squad sized unit into the SE1 portion of the panel. This unit will be assigned a SE assault position and will be the primary supporting unit for the ME assault on the objective.

- SE2: From the Unity3d Hierarchy window, select and drag a squad sized unit into the SE2 portion of the panel. This unit will be assigned a SE assault position and will be the secondary supporting unit for the ME assault on the objective.

B. USER INPUTS

- Sector Width: The angle in degrees that defines the size of the assault position sectors. Larger angles result in more nodes being considered for potential assault positions. Default is 20 degrees.

- Minimum Assault Distance (Min Asslt Dist): The lower bound of the assault position sectors. This is the closest distance a node can be to the enemy objective and still be considered for a potential assault position. Default value is 75 meters.

- Maximum Assault Distance (Max Asslt Dist): The upper bound of the assault position sectors. This is the furthest away a node can be to the enemy objective and still be considered for a potential assault position. Default value is 350 meters.

C. ASSAULT POSITION WEIGHTS

- Path weight (pathWeight): Value between 0–1 that represents the importance of the path cost in the total cost calculation of all potential assault positions.
- Combined Arms Weight (combinedArmsWeight): Value between 0–1 that represents the importance of the combined arms cost in the total cost calculation of all potential assault positions.
- Risk Value Weight (riskValueWeight): Value between 0–1 that represents the importance of the risk value cost in the total cost calculation of all potential assault positions.
- Assault Distance Weight (assaultDistWeight): Value between 0–1 that represents the importance of the assault distance cost in the total cost calculation of all potential assault positions.
- Neighbors Weight (neighborsWeigh): Value between 0–1 that represents the importance of the neighbors cost in the total cost calculation of all potential assault positions.

D. ATTACK POSITION WEIGHTS

- Path Weight (pathWeigh): Value between 0–1 that represents the importance of the path cost in the total cost calculation of all potential attack positions.
- Neighbors Weight (neighborsWeight): Value between 0–1 that represents the importance of the neighbors cost in the total cost calculation of all potential attack positions.
- Observer Weight (obsWeight): Value between 0–1 that represents the importance of the observer cost in the total cost calculation of all potential attack positions.

- Directness Weight (directnessWeight): Value between 0–1 that represents the importance of the directness cost in the total cost calculation of all potential attack positions.

APPENDIX C. VISIBILITY GRAPH EDITOR

This appendix provides additional details on each portion of the VisibilityGraph editor (Figure 47). Note that this appendix will only include inputs from the "Graphs" section of the editor. Additional details can be found in [2], Appendix B, Section B.

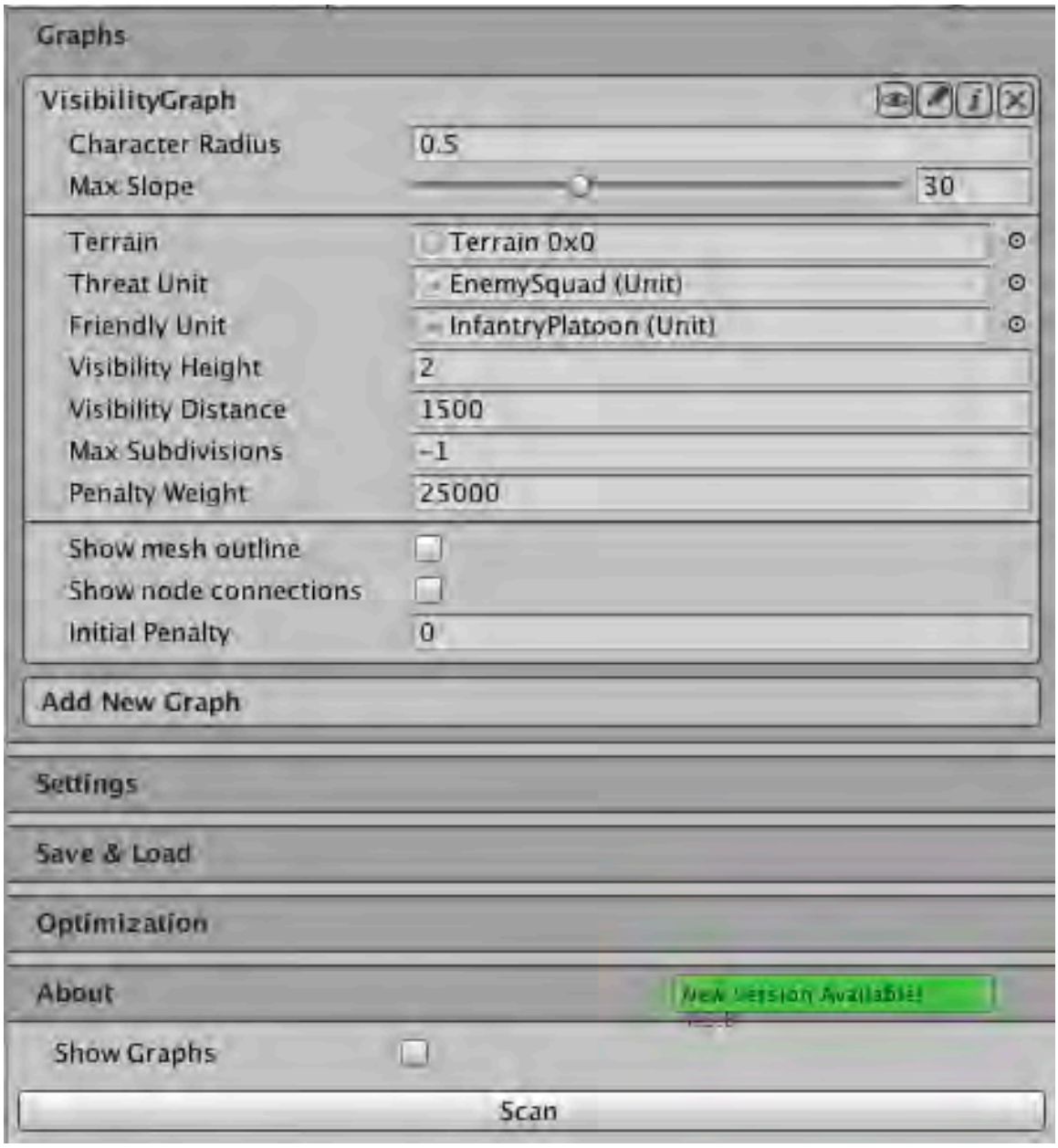


Figure 47. VisibilityGraph Editor

- Character Radius: A cylindrical radius that surrounds any agent that will navigate the VisibilityGraph [2].
- Max Slope: The max slope an agent is able to traverse [2]. Nodes with a gradient greater than the max slope are tagged as unwalkable.
- Terrain: Real-world imported terrain found in the Unity3d Hierarchy window [2].
- Threat Unit: The top-level enemy unit found in the Unity3d Hierarchy window [2]. In this case, the threat unit is an enemy squad.
- Friendly Unit: The top-level friendly unit found in the Unity3d Hierarchy window. In this case, the friendly unit is an infantry platoon.
- Visibility Height: The maximum agent height [2].
- Visibility Distance: The maximum distance considered for risk and observation annotations [2]. Default is set to 1500 meters for this system.
- Max Subdivisions: Default set to -1 [2]. Not intended to be adjusted.
- Penalty Weight: Scaling factor used to increase the importance of risk in path construction [2]. Default is set to 25,000.
- Show Mesh Outline: Used to visualize the navigation mesh outline [2].
- Show Node Connections: Used to visualize each node connection on the VisibilityGraph [2].
- Initial Penalty: Default set to 0 [2]. Not intended to be adjusted.

LIST OF REFERENCES

[1] *COMBATXXI Users Guide*. United States Army, Futures Command, White Sands, NM (TRAC-WSMR). [Online]. Available: https://cxxi.wsmr.army.mil/confluence/display/CXXIDOC/Home. Accessed: May 2, 2019.

[2] B. R. Harder, "Automated battle planning for combat models with maneuver and fire support," Ph.D. dissertation, Dept. of Comp. Sci., Naval Postgraduate School, Monterey, CA, USA, 2017.

[3] D. Miller, "Hierarchical task network prototyping in Unity3D," M.S. Thesis, Dept. of Comp. Sci., Naval Postgraduate School, Monterey, CA, USA, 2016.

[4] Unity Technologies. 2019. Unity3d, ver. 2019.1.0f2. [Online]. Available: https://unity.com/

[5] *Operations*, Marine Corps Doctrinal Publication 1–0, Headquarters Marine Corps, Washington, DC, 2017.

[6] *Planning*, Marine Corps Doctrinal Publication 5, Headquarters Marine Corps, Washington, DC, 2018.

[7] *Tactics*, Marine Corps Doctrinal Publication 1–3, Headquarters Marine Corps, Washington, DC, 1997.

[8] *Intelligence Preparation of the Battlefield/Battlespace*. Marine Corps Reference Publication 2–10B.1, Headquarters Marine Corps, Washington, DC, 2015.

[9] *Commander's Tactical Handbook*, Marine Corps Reference Publication 3–30.7, Headquarters Marine Corps, Washington, DC, 2016.

[10] *Infantry Platoon and Squad*, Army Technical Publication 3–21.8, Department of the Army, Washington, DC, 2016.

[11] *Marine Rifle Squad*, Marine Corps Reference Publication 3–10A.4, Headquarters Marine Corps, Washington, DC, 2018.

[12] A. Granberg. 2016. The A* Pathfinding Project, ver. 4.2.7. [Online] Available: https://arongranberg.com/astar/

[13] M. Jack, "Tactical Position Selection: An Architecture and Query Language," in *Game AI Pro: Collected Wisdom of Game AI Professionals*, S. Rabin, Eds. Boca Raton, FL, USA, 2014, pp. 337–359.

[14] D. Graham, An Introduction to Utility Theory," in *Game AI Pro: Collected Wisdom of Game AI Professionals*, S. Rabin, Eds. Boca Raton, FL, USA, 2014, pp. 113–126.

[15] R. Straatman, W. van der Sterren, and A. Beij, "Killzone's AI: Dynamic Procedural Combat Tactics," in *Game Developers Conference Proceedings*, San Francisco, CA, 2005.

[16] K. Xu, L. Sun, L. Qin, and Q. Yin, "Optimal Multi-objective Urban Tactical Position Selection," in *Proceedings of 2015 IEEE*, Beijing, China, 2015.

[17] W. van der Sterren, "Tactical Path-finding with A*," in *Game Programming Gems 3*, M. DeLour, Eds. Boston, MA, USA, 2002, pp.294-306.

[18] "Part 3: Pathfinding," class notes for Cognitive and Behavioral Modeling for Simulations, MOVES Institute, Naval Postgraduate School, Monterey, CA, summer 2018.

[19] A. Ilachinski, *Artificial War: Multiagent-Based Simulation of Combat*. Singapore: World Scientific Publishing Co., 2004.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California