SWBF ©

A TACTICAL SIMULATION

"The Devil is in the Details"

Table of Contents

I. Introduction

The simulation discussed in this paper is one created from scratch, using a customized engine that simulates a battle between two armies on a battlefield. The purpose of the simulation, named sWARm, is to determine how armies behave given various attributes and battle variables. This paper examines the effects of army composition, vision, and initial army distribution on the outcomes of the battle simulations.

In sWARm, each battle is composed of two armies, with each army containing various units with specific attributes. Due to the random nature of the war, two battles with the exact same starting parameters can result in a different result each time, since many decisions that are made are random. Over many successive runs however, aggregate values which are more useful for analysis can be determined. A single run of sWARm, while entertaining to watch, is not indicative of a successful army composition. As such, sWARm is meant to be run in a batch mode, where aggregate results are calculated over 1,000 runs of the same initial battle conditions. The sWARM simulation in non-graphic batch mode (for multiple runs and calculations), command-line simple graphics mode, or full GUI interface graphical mode.

II. Agent Attributes

The agents in a sWARm simulation are called units. In order to make the agents in the sWARm simulation behave as realistically as possible, it is important to take into account the variable statistics most relevant to a battle that would influence a unit's behavior. The following is a brief description of each unit attribute.

Hit Points – This attribute denotes the health of a unit. The real equivalents of hit points are life, health, and wellness of a soldier or vehicle. When damage is dealt to a unit, the damage is subtracted from its remaining hit points. If at any time a unit's hit points reach 0, the unit is deemed dead. Hit points do not regenerate and once hit points are lost, there is no way to regain them. For infantry units, this attribute is normally distributed with a small standard deviation. For mechanical units, this attribute is constant.

Armor Coverage – This attribute denotes the probability that a unit's armor will mitigate the damage dealt to it. The real equivalent of armor coverage is the percentage of the body of a unit or vehicle that is covered by armor, so that a fired shot has a chance of hitting armor or the bare skin of a body or an exposed part of a vehicle. If armor coverage is denoted by a probability p, the probability that damage dealt to a unit is *mitigated*, is p and the probability that a unit will incur the full damage from the attack is $1 - p$. This probability p is set higher for units with sturdy armor and protection and lower for units that do not rely on close combat. For mechanical units, this attribute is relatively high for armored vehicles and low for non-armored vehicles.

Armor Effectiveness – This attribute denotes the percent of the total damage dealt, that the unit will incur if the armor mitigates the hit. The real world equivalent of this armor effectiveness is the thickness or toughness of the armor possessed by a unit or vehicle. If armor effectiveness is denoted by a probability *p* and damage dealt is denoted *d*, the amount of damage incurred by the unit is (1-*p*)**d* and amount of damage prevented is *p***d*. This probability *p* is set higher for units with thicker armor, and lower for units that do not specialize in close-quarters combat. For mechanical units, this attribute is high for armored units and low for non-armored units.

Vision – This attribute is a quantifier for how far a unit can see in any direction, with modifiers when it is looking up a hill or down a trench. The real world equivalent of vision is the vision of a soldier or the aiming mechanism for a tank. This attribute is very high for snipers and units with technological aiming techniques and low for regular human units.

Splash Damage Factor – This attribute defines the percentage of normal attack damage that will be dealt to any adjacent units to the unit being targeted. This can be thought of as shrapnel, or an explosion that spreads to a large area. If the splash damage factor is denoted as *p*, and the damage it regularly deals is denoted *d*, the damage dealt to each unit in a space adjacent to the attacked space is *p***d*. Units that fire guns do not have this capability, but units that fire grenades, explosives, or shrapnel have the ability to harm units outside their immediate attack space.

Base Damage – This attribute is the average amount of damage a unit does per attack. This can be thought of as the amount of damage a gunshot, grenade, or explosive shell does to an enemy in a battle. Along with distance and armor modifiers, this attribute determines how much damage will be done to an enemy in a given round of the simulation.

Damage Standard Deviation – This is the standard deviation of the amount of damage a unit does per attack. Not every attack can do damage equal to the base damage since that would be unrealistic. This can be thought of as imprecise hits, excellent hits, or modifications of a normal hit in battle. For attacks like gunshots, the standard deviation is relatively small, since the range of damage a gunshot does is not extremely high. However, for explosives and shrapnel, the standard deviation is much greater since these types of attack are inherently more random and volatile by nature.

Maximum Actions Per Turn – This is the maximum number of moves and/or attacks per turn. This attribute is analogous to a vehicle or soldier's speed in battle. Infantry are fast units and can achieve more decisions per turn, while tanks and snipers are slower and more calculating, which means their decisions are slower, and the maximum number of decisions that they can make per turn is much lower.

Probability of Multiple Actions Per Turn – The probability of a unit being able to have "another decision" in a single turn. Each unit will be allowed one decision per turn and each successive decision must meet this probability threshold in order for the unit to be

allowed to make the extra decision. If a unit has a multiple action probability *p*, it will make a decision and it's probability of making a second move will be *p*, and each successive probability will be reduced by a factor of *p*. For example, the fourth turn will only have a *p*^3 probability of occurring. This compounding is done until the maximum actions per turn threshold is reached, at which time a unit is not allowed any more decisions for the turn.

Accuracy – This attribute denotes the probability that an attack will hit another unit. The real equivalent of accuracy is accuracy of guns and precision of the weapons of a soldier or vehicle. If a unit's accuracy is denoted by *p*, the probability that an enemy unit will be hit by the unit's attack is *p*^*d*, where *d* is the distance between the two units.

Speed – This attributes determines which units act first in a turn and is used for determining the maximum number of decisions per turn.

Maximum Slope Traversable – The maximum height a unit can travel in one move. This equates to a unit on foot having more mobility than a treaded vehicle.

III. Unit Types

sWARm armies currently consist of up to 200 units comprised of up to four main unit types, each with different characteristics and behavior patterns. The unit types are infantry, sniper, tank, and mortar. The specifics of the units will be described below.

A. Infantry

"These units are the lifeblood of an army. They are fast and well armored for their size. While an infantryman's accuracy and damage may not be overly impressive, his speed compensates for this deficiency. An infantry unit is usually able to make several movements or attacks per turn, and thus is usually the first to arrive at a battle, and creates a formidable fighting force when combined with several other infantrymen."

Hit Points	Normally Distributed (μ =15, σ = 2)
Armor Coverage	70%
Armor Effectiveness	20%
Vision	50%
Splash Damage Factor	0
Base Damage	7
Damage Standard Deviation	2
Maximum Actions/Turn	5
Probability of Multiple Actions/Turn	75%
Accuracy	75%
Speed	20
Maximum Slope Ascent/Descent	3

Infantry Attribute Table

B. Sniper

"Snipers are painfully precise, and very deadly. An opponent caught off guard by a cadre of snipers will certainly regret his lack of foresight, as they will pick off infantrymen from a distance with little difficulty. A sniper's weakness lies in his speed and close combat skills. He is usually only able to make one to two attacks or movements per turn, and thus a battle can be lost or won before snipers are able to come to their army's aid. Additionally, low armor levels for snipers make them easy targets in close-combat."

Hit Points	Normally Distributed (μ =13, σ = 1)
Armor Coverage	55%
Armor Effectiveness	10%
Vision	100%
Splash Damage Factor	0
Base Damage	20
Damage Standard Deviation	1
Maximum Actions/Turn	2
Probability of Multiple Actions/Turn	50%
Accuracy	95%
Speed	5
Maximum Slope Ascent/Descent	3

Sniper Attribute Table

C. Mortar

"These units certainly provide plenty of "bang" for your buck. They move significantly faster than tanks, and offer immense amounts of damage to a designated target along with splash damage to the area surrounding a target. With the approximate strength of an infantryman, a mortar-man is versatile, and several of them can often take an opponent by surprise with devastating consequences. Unfortunately, their low hit points make them vulnerable to enemy tanks and snipers."

Hit Points	Normally Distributed (μ =15, σ = 2)
Armor Coverage	50%
Armor Effectiveness	20%
Vision	65%
Splash Damage Factor	10%
Base Damage	20
Damage Standard Deviation	3
Maximum Actions/Turn	3
Probability of Multiple Actions/Turn	55%
Accuracy	55%
Speed	10
Maximum Slope Ascent/Descent	2

Mortar Attribute Table

D. Tank

"A tank is the workhorse of an army. This unit will usually outlast most others for two reasons: it possesses an incredible number of hit points, and it will be the last to reach the battle. The tank's lack of speed is balanced by its range and destructive firepower. One shot from a tank will not only shatter its target, but will severely cripple all units in the surrounding area. Although tanks may not get to take as many shots as other units, the ones they do take count for quite a bit."

Hit Points	60
Armor Coverage	90%
Armor Effectiveness	40%
Vision	80%
Splash Damage Factor	20%
Base Damage	20
Damage Standard Deviation	6
Maximum Actions/Turn	1
Probability of Multiple Actions/Turn	0%
Accuracy	65%
Speed	1
Maximum Slope Ascent/Descent	1

Tank Attribute Table

IV. Decision-making Algorithms

One of the most important steps in creating the simulation is determining how an individual unit will establish what decisions it needs to make, and the steps required for making the decisions. The sWARm simulation uses a multi-layered decision-making process which makes an agent determine first whether it will attack or move. If an agent decides to attack, the attack algorithm will be invokes, and if it moves, the move algorithm will be invoked. Each of these algorithms is *specific* to each unit type to closely mimic their predicted behavior in a battle. One of the critical factors used in making a decision is the vision of the agent as the number enemy and friendly agents it sees will ultimately determine what action it takes.

A. Vision Algorithm

An integral part of an agent's decision and movement process is its vision. This aspect of the simulation works quite unlike similar features of other models and frameworks. Models such as Schelling's model take only adjacent or once removed spaces into account when determining the visible points. The algorithm for the sWARm simulation is based on the altitude of terrain surrounding any point in question.

To calculate the overall visible range from a point, the maximum vision in each of the cardinal directions is obtained. This is done by iterating in each direction until a boundary is reached, a maximum height differential is met, or the maximum vision is reached. It should also be mentioned that a unit on one side of a hill cannot see a unit on the other side, which makes the simulation more realistic. The number of iterations before one of these conditions is met is recorded as the maximum vision for that cardinal direction from that point. After these numbers are recorded they are combined by concatenating the area of the polygon formed by them and the area achieved by using the

Java API's quadTo() function. This gives the effect of offering a rounded area of visible points that matched what would realistically be visible in similar terrain conditions. In total, there is a list of visible points created for each army, for each unit type, and for each point on the map. While at first this may seem excessive and wasteful, it is done to minimize the amount of in-simulation calculations required per turn.

Each unit type has a vision modifier, that acts as a handicap on the vision in each of the cardinal directions before the polygon and quadTo() calculations. This limits the vision of certain unit types, having a profound effect on the outcome of the simulation. During the analysis of data produced by the simulation, it was discovered that an agent's vision is one of the most critical factors in determining number of kills as well as probability of living through the battle. This can be most profoundly seen when taking the example of a sniper vs. an infantry unit. While the infantry can move or attack up to five times per turn, they hardly ever come within close enough range to provide competition to a pack of snipers. The enhanced vision range of the sniper allows it to keep other units at a safe distance, a key factor for its survival due to its slower movement.

B. Attacking vs. Moving

The first consideration in creating a decision structure was to create a way for a unit to determine whether it should attack or move. To that end, several lists of possible things to consider when making this decision were created. One factor that seemed consistently and realistically important in a battle situation was the ratio of friends to enemies in a unit's vision. In other words, if a unit can see one enemy and five friends, it probably means that the unit is too far from the battle to be effective. Therefore, it should

move. By the same token, if a unit can see several enemies, but no friends, it should move due to a high probability of hostile fire without "backup". If the ratio of enemies to friends is even, or within a reasonable range of even, a unit will decide to attack because it is both close to the battle (presumably) and has sufficient fire support. This way, a unit will only attack if it has a reasonable chance of victory, survival, and making a contribution to the main battle rather than getting tied up in small skirmishes. In this way, the unit is still thinking for itself, while still contributing to an overall army strategy.

To execute this decision strategy, a unit returns an ArrayList of points that are within its vision. Each point found is then examined on the map to determine whether it is occupied, and if so, whether it is occupied by a friend or enemy unit. The number of enemies and friends is totaled, and a ratio is calculated. If the unit has a reasonable enemy to friend ratio, it will attack. Otherwise it will choose to move.

C. Movement Algorithm

Once a method for determining whether a unit ought to move or attack was created, the next decision a unit needs to make is to determine where it will move to. Since the agents in the simulation have two major priorities, staying alive and killing other units, it is easy to determine that a unit should try to move to a space that brings its visible enemy to friend ratio closest to 1:1. Thus, first the valid moves a unit could make (determining which of its neighboring spaces wasn't occupied or out of bounds) is determined. The visible friend to enemy ratio from every point that a given unit can feasibly move to is determined. A unit would thus move to the space with the best ratio (closest to 1:1).

It seems logical that if a unit finds two spaces to offer it the same enemy to friend ratio, that it would try to move to the space that increased its vision more. This can either be accomplished by moving further from the border of the map, or more often, by moving to a space of greater height. The space to move to was determined by generating the total potential number of points visible for each space in question and then moving to the space with the largest number of visible points.

In order for the units to move into the range of the enemy army, an initial bias is needed. This bias is based on a Bernoulli random variable that is generated at the runtime of each move. By this bias, a unit would try to move toward the enemy army location if it did not have any enemies in sight. We rationalized the unit's knowledge of its enemy's location with the idea that in a real battle, any army engaging in battle would have *some* idea where to go in order to engage its enemy. Once this was added to the movement algorithm, the armies appeared to move toward each other as expected.

D. Attack Algorithm

The way one unit type attacks is quite different from the way another unit type attacks. Each unit type has a distinct attack algorithm. The attack algorithms are described in detail in this section.

i. Infantry

An infantry unit's main objective is to weaken the enemy's front line. In order to do this, all infantry units will always go for the quickest kill. This is determined by calculating the potential target's estimated hit points. The estimated hit points is a value equal to that unit's current hit points minus the estimated damage that the attacking unit can deal. This test is performed on all of

the enemy units that fall within the infantry's attack range, and the unit with the lowest estimated hit points that is below 0 will be attacked.

If there is no enemy unit within range that is estimated to be killable in one shot, the infantry will attack the most "dangerous" unit within range. A unit's threat level is computed by multiplying its hit points, accuracy, and the average damage dealt per attack, all divided by the current distance from the attacking unit.

ii. Mortar

The mortar is a specialized unit meant for decimating slower, more heavily armored units (such as the tank), and as such, its first attack priority is to scan its attack range for the unit with the highest armor cover times armor effectiveness. If there is a tie, it will attack the most dangerous unit within its vision range.

iii. Sniper

Similar to the infantry unit, a sniper will compute the estimated hit points for all enemy units within its expansive attack range. Of all the units for which this value is less than zero, the sniper will attack the unit with the highest hit points that it still believes it can kill in one shot. This is equivalent to the sniper attempting "headshots" at the most powerful enemy units.

If there is no unit that is killable in one shot, a sniper will check to see if there are any units within a range of 3 tiles. If there are units within 3 tiles, the sniper will attack the most dangerous unit, as computed above. If there is no

enemy unit within 3 tiles of the sniper, the sniper will attack the unit that is closest to the 3 tile radius, breaking ties with the danger calculation.

iv. Tank

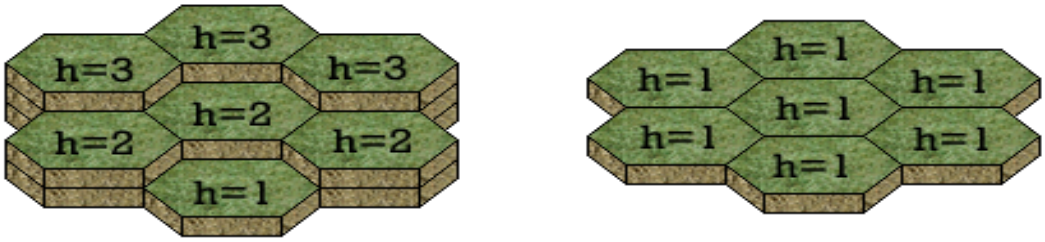
The primary benefit of the tank as an attacking unit is its large area of splash damage. Splash damage allows for the tank to kill more than one enemy unit per attack. A tank estimates the ratio of total damage dealt to enemy units and friendly units for each attacking tile. The tank will only attack a unit if the estimated splash damage that it deals to adjacent squares injures enemies at a higher ratio than it injures friends.

If the tank cannot find a unit to attack such that the ratio of the expected value of damage done to enemies is above the designated level (default 1) then the tank will not fire at all and will choose to move instead.

V. Maps and Positioning

A. The Map

The simulation maps are arranged in a hexagonal grid, 50 tiles high and 50 tiles wide, with each tile on the map being adjacent to 6 other tiles. The traditional 2x2 quadrilateral grid is too simplistic and would not mimic battle movement as well as a hexagonal grid. A unit can move north, south, northwest, southwest, northeast, and southeast in the sWARm simulation. Each map has a corresponding height map that is a grid of integers that represents the heights for each space on the map. This height map is used for vision, movement, and attack calculations during the simulation.



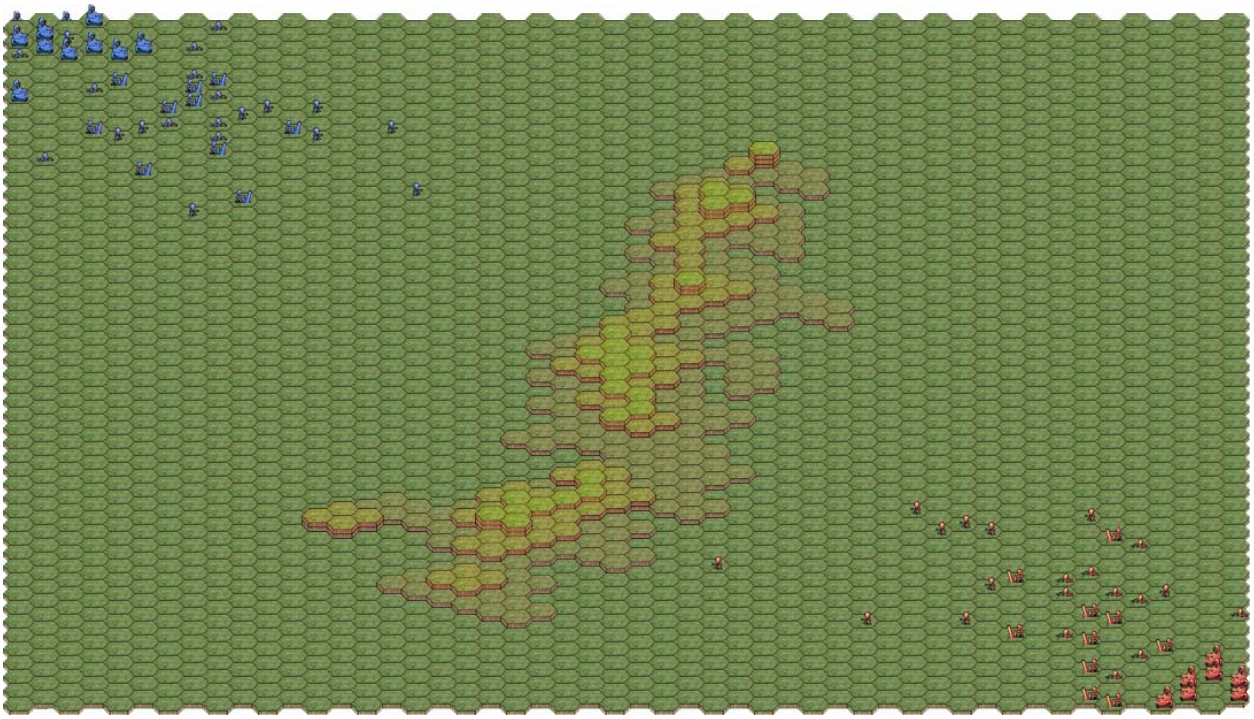
Visual Example of Map tiles with corresponding height values.

B. Initial Positioning

In an earlier version of the sWARm simulation, units were dropped into random positions on the map when they were created. It was determined that in a real battle, it is far more likely that units in the same army will start together, rather than being dropped randomly as if the opposing armies literally fell from the sky. With that in mind, the default position for the "first" army was set to the northwest corner of the map, and the "second" army to the southeast corner of the map. The units are randomly placed "close"

to a certain corner point corresponding to which army they are in. It was while tinkering with the starting positions that it was determined that perhaps the starting positions of units within their armies should be slightly less random. It is logical that the faster units ought to be biased to start toward the front of the army (so that they can get into skirmishes early) and the slower units should be biased toward the rear of the army. This was accomplished by multiplying the randomly generated x and y starting values for a unit by its speed divided by its army's average speed (Unit Speed/Avg. Army Speed).

When attempting to put the armies in their respective corners, it was discovered that there were far too many "collisions" being generated by the random function (units attempting to start where another unit had already been placed). Thus, a modification was made by creating a customized Random class that gave a *normally distributed* random value. This class took both an average for the random value, and a standard deviation as parameters.



C. The "Spread" Value

The sigma, or "spread" value of an army determines how far apart an army's units are at the start of the battle. This value, discovered while creating values for starting positions of units, is actually the standard deviation of a normally distributed random variable. It determines how much a random value can "deviate" from its given average value. Thus, when sigma is small, an army's units will be very close together, since they will be more likely to be placed in a position that is close to the average starting point. When sigma is large, however, an army will be very spread out. It was originally thought that being able to determine how spread out an army would be was simply a nice feature to have. However, after running a few movement simulations, it was discovered that the sigma is quite important in determining how well an army does. Due to the decision and movement rules implemented, pitting armies with different spread values against one another creates very different results. An army with a larger spread value, for example, is more likely to attempt to surround an army with a smaller spread value. This flanking behavior would, all else being equal, allows an army with a larger sigma to triumph over an army with a smaller sigma. Additionally, it is more likely that too large an initial spread would make an army easy to kill, since its units could be picked off one by one. Pitting two armies with large spread values against one another results in many small skirmishes, whereas two armies with small spread values usually collide in one huge battle.

VI. Statistical Analysis Tracker:

An integral part of the sWARm simulation is the built in statistical tracker that allows easy analysis of data after a battle has completed. The statistical tracker was created using a class that keeps track of all important events in the battle while the events occur. For example, whenever a unit is killed, the tracker notes the type of unit killed, the type of unit that killed it, and the damage dealt to the unit.

The following is a sample data table created by stat tracker and formatted for readability. The table shows many statistics that are important in determining the events that transpired during the simulation. The beauty of the statistical tracker is that it is easily extensible and numerous other statistics and ratios can be calculated.

Steps for Win	80	**Army 1 Infantry Damage Dealt**	129
Winning Army	1	**Army 2 Infantry Damage Dealt**	182
Army 1 Soldiers Remaining	14	**Army 1 Mortar Damage Dealt**	132
Army 2 Soldiers Remaining	1	**Army 2 Mortar Damage Dealt**	112
Army 1 Initial Size	40	**Army 1 Sniper Damage Dealt**	233
Army 2 Initial Size	40	**Army 2 Sniper Damage Dealt**	286
Army 1 Infantry Left	0	**Army 1 Tank Damage Dealt**	212
Army 2 Infantry Left	0	**Army 2 Tank Damage Dealt**	222
Army 1 Mortar Left	0	**Tank Moves**	667
Army 2 Mortar Left	0	**Sniper Moves**	876
Army 1 Sniper Left	8	**Mortar Moves**	902
Army 2 Sniper Left	1	**Infantry Moves**	1329
Army 1 Tanks Left	6		
Army 2 Tanks Left	0		

Sample Statistical Tracker Data
(Based on 10-10-10-10 armies pitted against each other)

Stat tracker is not very useful in terms of single runs, but in conjunction with batch mode, it is a powerful tool for data analysis. The current version of the code does not allow direct access to the statistical tracker except through comma delimited text files produced in batch mode, which is what the data in the following analysis sections are based on.

VII. Unit Distribution Analysis:

Unit distributions have a profound effect on the behavior of an army on the battlefield. Homogenous armies have very specific strengths and weaknesses. The following is an analysis of the attributes and behaviors of a homogenous army consisting of all of the same unit type.

A. Homogenous Distributions

i. Infantry

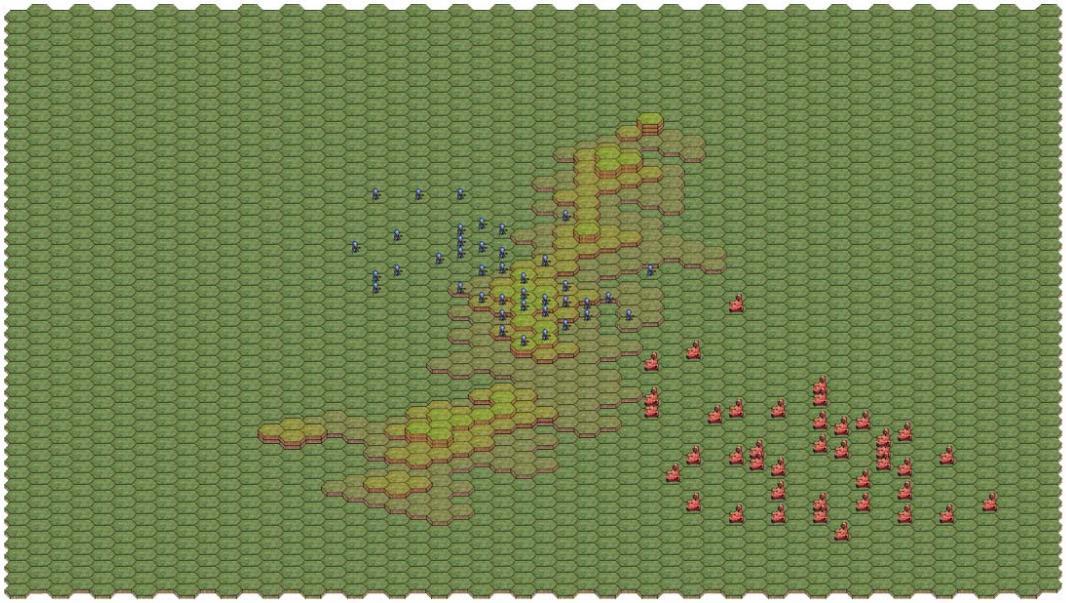
The advantages of an infantry army are plentiful. All the units are fast, which means that they reach opponents quickly, attack more per turn, and move more per turn. In terms of a real world model, the infantry units would be the least costly of any of the units in this simulation. The disadvantages of an infantry army are that the hit points of the units are relatively low and most attacks by any other unit can kill the unit with one attack. Additionally, another factor that cripples infantry armies against tough adversaries is the low vision modifier of infantry units. Infantry units do not possess the range of a mortar, a sharp-shooting sniper, or a high tech tank.

When pitted against homogenous armies of the same number of mortars, snipers, and tanks, infantry armies were often defeated. Tank armies deal too much damage for infantry to overcome, and the splash damage dealt by a tank shell is often enough to kill an infantry unit. This is the case for mortars as well, but infantry armies often did much more damage to mortar armies because of the low hit points of the mortar units. Finally, in a battle against sniper units, infantry were demolished. The sniper's combination of range and accuracy are simply too powerful for infantry units to overcome.

Though the infantry often lost battles to homogenous armies of the same number of units, this is quite unrepresentative of their true performance in battle. Infantry units are the most cost efficient units in an army and most of the time, the number of infantry is 2, 5, or even 10 times more than the number of mortars, snipers or tanks.

Army A	Army B	Army A Win %	Army B Win %
40 Infantry	40 Mortars	6.3%	93.7%
40 Infantry	40 Snipers	0.0%	100.0%
40 Infantry	40 Tanks	0.0%	100.0%
40 Infantry	20 Mortars	28.5%	71.5%
40 Infantry	20 Snipers	20.1%	79.9%
40 Infantry	10 Tanks	15.4%	84.6%

Table: Infantry win % versus other homogenous army types



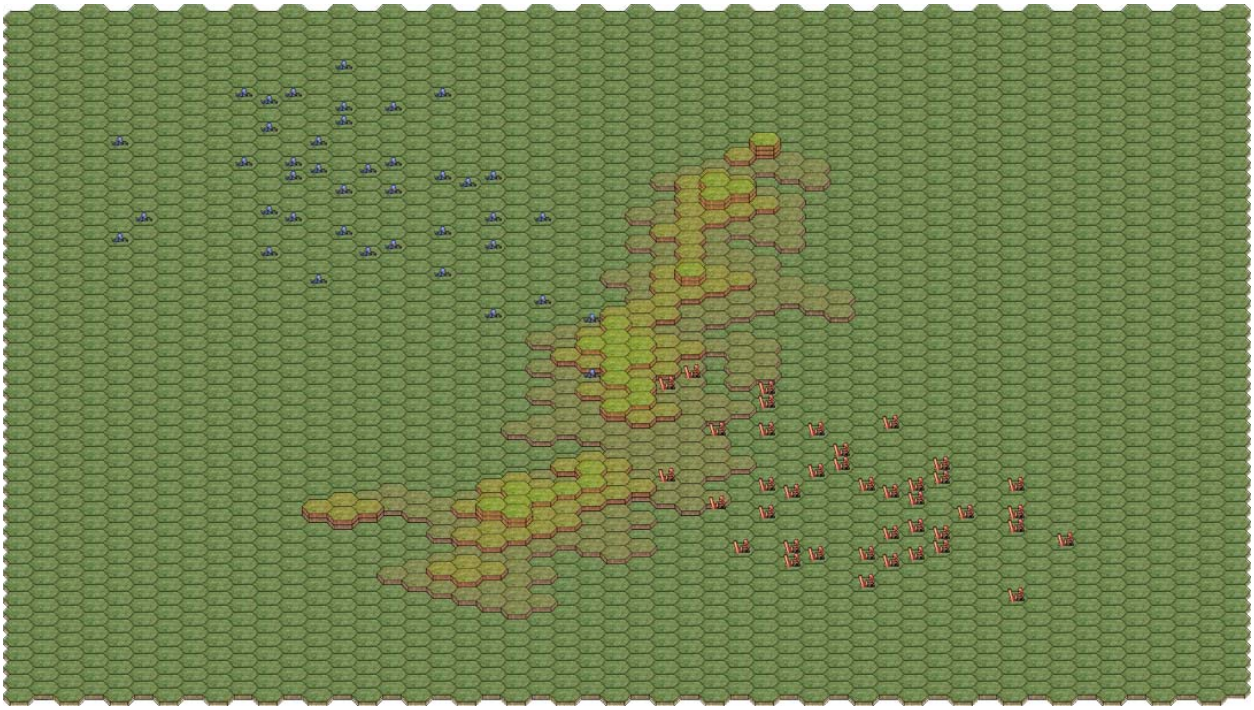
ii. Sniper

The range and accuracy of a sniper is unparalleled. The snipers can kill many units before they enter its range and often hits its enemies. In terms of a real world model, the sniper units would be rather expensive since their equipment is very high tech and the training needed to produce a high quality sniper requires large investments. Though snipers have a high accuracy and range, if units get in close to a sniper, the sniper's attack range advantage is decimated. With low coverage and effectiveness, any unit that hits a sniper will often kill it. Additionally, due to the time it takes to set up the gun and make the precision attack, the speed of a sniper is relatively slow.

When pitted against homogenous armies of the same number of mortars, infantry, and tanks, the snipers prevailed against the infantry often, and often were defeated by the tanks usually, but not always, were defeated by mortars. As with infantry, the snipers cannot handle the immense amount of damage tanks do in terms of splash damage. Multiple tanks firing multiple shells quickly kill snipers if the snipers allow the tanks to get in range. Snipers usually lose to mortars because of their splash damage, but sometimes win if they get lucky and pick off a few mortars early. This happens with mortars and not tanks because mortars have a low hp, so a low probability attack that hits, may kill a mortar, when it would only slightly damage a tank.

Army A	Army B	Army A Win %	Army B Win %
40 Snipers	40 Infantry	100.0%	0.0%
40 Snipers	40 Mortars	30.8%	69.2%
40 Snipers	40 Tanks	0.0%	100.0%
20 Snipers	40 Infantry	79.9%	20.1%
40 Snipers	30 Mortars	50.8%	49.2%
40 Snipers	20 Tanks	25.5%	74.5%

Table: Sniper win % versus other homogenous army types



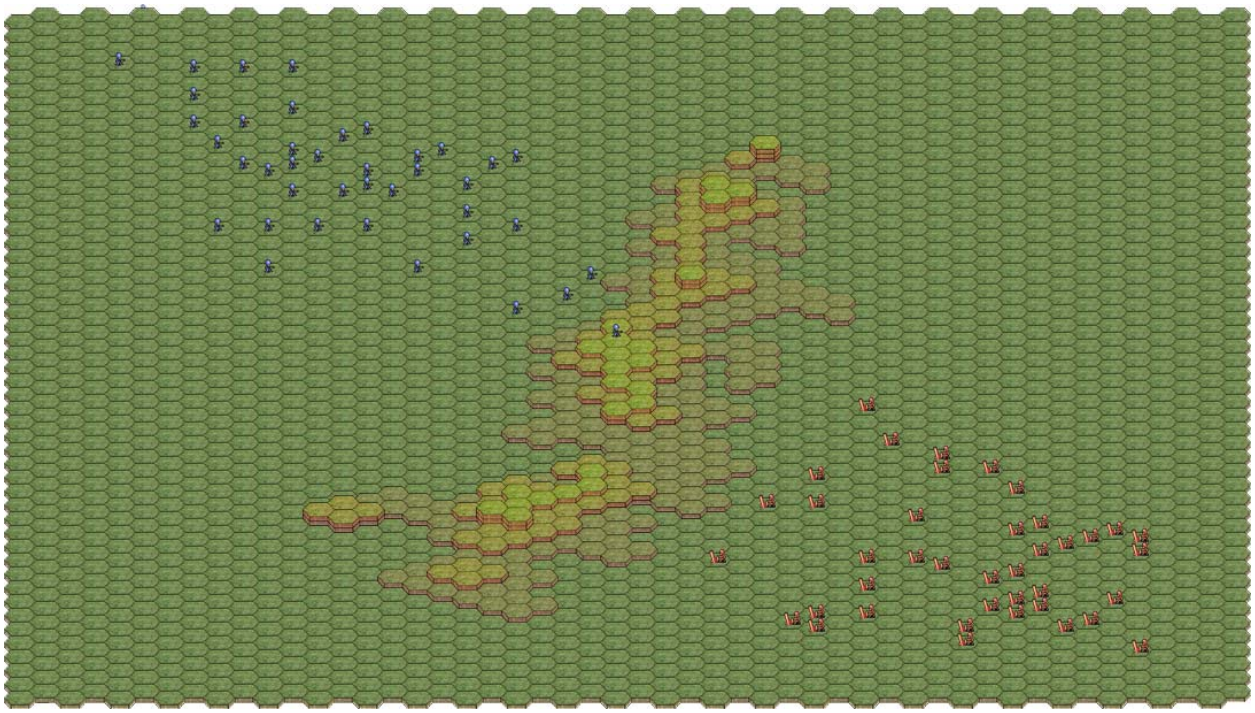
iii. Mortar

The mortar is a unit that does the damage of heavy artillery, but is not nearly as slow. The mortar can make multiple attacks per round (2-3 on average), do as much damage as a tank, and has comparable accuracy. Mortar units are relatively inexpensive in the real world, in comparison to tanks and snipers but are slightly more expensive than infantry units because the mortar machinery is relatively complex. Unfortunately, mortar units have low hit points, do less splash damage than tanks and don't have the explosive capability of tank shells, meaning that the units are easier to kill and do much less damage than a tank.

When pitted against homogenous armies of the same number of infantry, snipers, and tanks, mortar armies performed on par. The interaction between mortar armies and sniper and infantry armies has already been discussed, but the interaction between mortar armies and tanks is very intriguing. The battles between these two army types are often very close and each army wins battles approximately 50% of the time. The reason tanks compete well with mortars is because tanks have high hit points so they are not easily destroyed. Additionally, tanks have high splash damage, which kills many mortars because of their low hit points. The reason mortars deal well with tanks is because they have a high attack rate and a high rate of damage, compounded with splash damage. These high damage attack turns allow mortars to kill tanks in 1 or 2 turns, when other units are unable to do so.

Army A	Army B	Army A Win %	Army B Win %
40 Mortars	40 Infantry	6.3%	93.7%
40 Mortars	40 Snipers	69.2%	30.8%
40 Mortars	40 Tanks	0.0%	100.0%
20 Mortars	40 Infantry	71.5%	28.5%
30 Mortars	40 Snipers	49.2%	50.8%
40 Mortars	20 Tanks	29.9%	70.1%

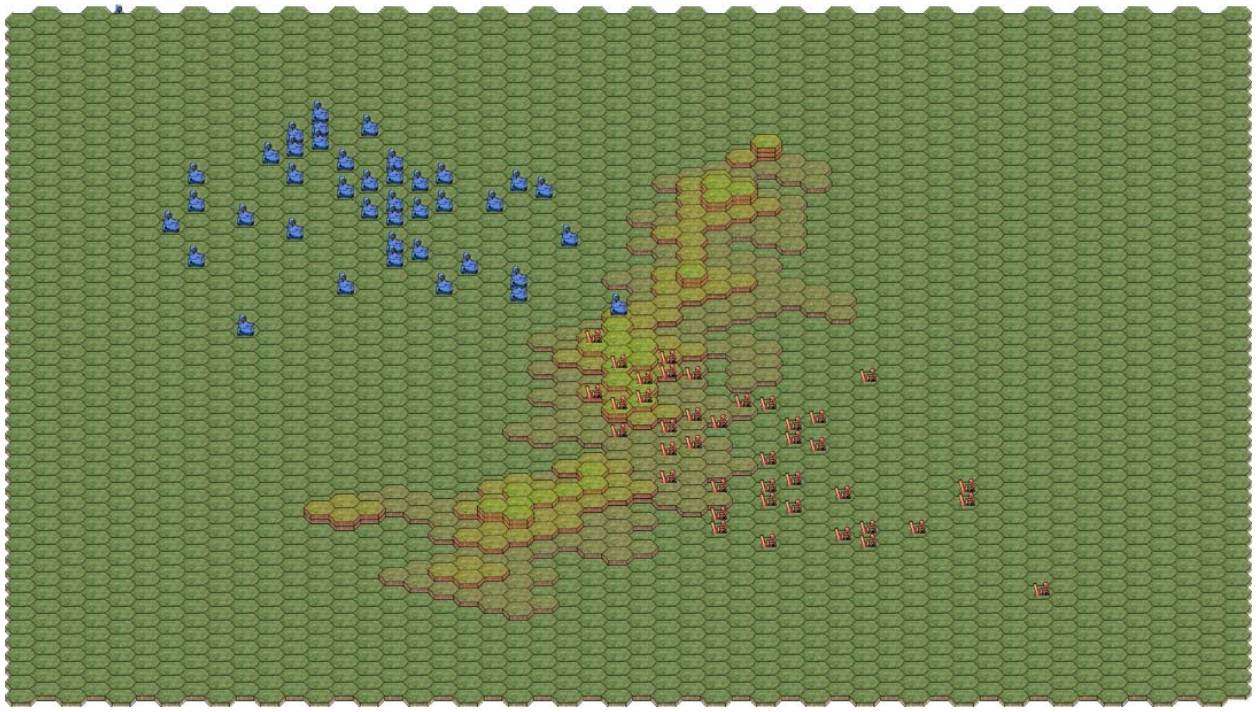
Table: Mortar win % versus other homogenous army types



iv. Tank

The tank is by far the most dominant force on the battlefield. When pitted against homogenous armies of the same number of infantry, snipers, and mortars, the tanks almost always prevailed. This can be attributed to the large amount of splash damage the tank does in conjunction with the high armor and high hit points of the vehicle. Only when the number of tanks is cut in half or reduced by a factor of 4 does the battle become more even.

This behavior is not totally unexpected since tanks are the most expensive of all the battle units. The cost for a single tank easily exceeds the cost of multiple infantry, snipers, and mortars.



Army A	Army B	Army A Win %	Army B Win %
40 Tanks	40 Infantry	100.0%	0.0%
40 Tanks	40 Mortars	100.0%	0.0%
40 Tanks	40 Snipers	100.0%	0.0%
10 Tanks	40 Infantry	84.6%	15.4%
20 Tanks	40 Mortars	70.1%	29.9%
20 Tanks	40 Snipers	74.5%	25.5%

Table: Tank win % versus other homogenous army types

B. Selected Heterogeneous Distributions

i. 40 Snipers vs. 35 Mortars/35 Infantry

Army A	Army B	Army A Win %	Army B Win %
40 Snipers	35 Mortars 35 Infantry	94.5%	5.5%

Battle Results

While developing unit types and attributes, it was concluded that a sniper could not simply be an everyday, mass-produced soldier. A sniper was to be "elite," in that a single sniper unit ought to be a serious challenge to multiple low-armor opposing units at range. This conclusion was reached when it was realized that in reality, a truly good sniper was not a common thing. Snipers go through years of training to effectively become killing machines, and are expected to provide a great deal of support to their team.

The sniper unit was tested against the two other "human" units. Because a sniper is made to be effective against flesh targets, introduction of the heavily armored tanks would not have given an accurate statistics on the true value of the unit. While snipers are deadly at long range, the combination of mortars and infantry yields a balanced and devastating mixture of speed and power. Thus, a batch of battle simulations, each consisting of 35 infantrymen and 35 mortars against 40 snipers was run. To ensure that the units' effectiveness was tested, each unit defaulted to a max vision of 4, and each army to a sigma of 5.

Apparently, the snipers were quite as effective as they were designed to be. The extraordinarily high winning percentage can be attributed directly to a sniper's expansive vision and exacting accuracy. The sniper army kills every unit attempting to get to the battle before they are close enough to respond. Once the attrition of the opposing army has begun, what ensues is practically a domino effect. Each unit of the opposition moves over his dead allies only to be mowed down next.

The (rare) loss of the sniper army occurs only when it is distributed such that the snipers do not start close enough together to effectively repel their opposition. Since snipers are relatively slow moving compared to infantry, starting the simulation with the army of snipers far apart from one another gives infantry sufficient time to get close enough to attack (and soon be backed up by mortar shells). Barring this scenario, however, snipers certainly prove their superiority over other human units.

ii. 50 Infantry/20 tanks vs. 45 Mortars/10 Snipers

Army A	Army B	Army A Win %	Army B Win %
50 Infantry 20 Tanks	45 Mortars 10 Snipers	15.6%	84.4%

Battle Results

This situation was examined because it presented an opportunity for testing of unit balance. In this battle, an aggressive force of fifty infantry and twenty tanks (army A) fight a very defensive force of forty-five mortars and ten snipers (army B). Army A is comprised of the very fast infantry units that often get multiple moves or attacks per turn. It also has the incredibly rugged and powerful tank, so as this force quickly advances it has the power to overrun oppositions. Army B is, however, an almost perfect compliment to this attacker. This force turns out to be defensive both due to the fact that it has the very long ranged snipers, and the mortars, which target tanks to protect the rest of the units.

In observing and analyzing the simulation it becomes clear that the advancing army of infantry is destroyed at range by the snipers leaving a protected path for the mortars to advance and eradicate the force of tanks. This battle strategy emerges each running of the simulation, even the times when army A wins. For army A to win, it is imperative that the force of infantry advance slowly, allowing the tanks to arrive at the battle with them, letting them overrun army B.

iii. 20 Mortars/20 Tanks vs. 100 Infantry

Army A	Army B	Army A Win %	Army B Win %
20 Mortars 20 Tanks	100 Infantry	42.8%	57.2%

Battle Results

In a battle of heavy artillery versus a large battalion of infantry, battle outcomes are very uncertain. In a very close battle, on average the 100 infantry prevail. The dynamic of this battle is very interesting in that as the armies enter the center, units die off really quickly on both sides. The reason for this is twofold. The first and most obvious reason for this massacre at the center of the map is the fact that mortars and tanks do extremely high amounts of damage and since the infantry units are often bunched together since there are so many of them, many infantry units incur splash damage from the tanks and mortars. The second reason for this huge massacre is the low hit points of the mortars in conjunction with the extra attacks that infantry units get per turn. With so many infantry on the map, when there are many opposing units at the center of the map, they will incur a very large amount of damage because the infantry units are all using multiple attacks. The outcome of this battle is largely determined by luck in the first few attack steps of the simulation. The reason for this is that if the tanks and mortars manage to kill enough infantry in their initial volley, the firepower left in the infantry army is rarely enough to take down the heavily armored tanks.

iv. 50 Infantry/10 Snipers vs. 40 Infantry/20 Mortars

Army A	Army B	Army A Win %	Army B Win %
50 Infantry 10 Snipers	40 Infantry 20 Mortars	27.2%	72.8%

An interesting discovery was the army strategy that formed between infantry and snipers on the same team. While snipers are powerful on their own, in combination with infantry, a deadly strategy is developed. Because snipers are inherently slower than most units, they arrive at the battle many steps after the attacks have begun. The inclusion of infantry in the armies allow the snipers safe distance and cover, while at the same time preventing enemy units from advancing. This allows the snipers easy access to weak enemy units, for maximum killing efficiency.

For this battle, armies of 50 infantry and 10 snipers were pitted against 40 infantry and 20 mortars. Because of the mortar unit's attack range, the infantry / sniper army's default strategy was effectively countered. Both infantry battalions reach each other and begin to exchange fire. Due to the mortar's speed advantage, they are able to form a line and set up before the snipers get within range. Based on the mortar's attack algorithm, they will attack the most dangerous units within their vision, which are the opposing infantry, that is, until the first sniper appears. The splash damage caused by the mortars damages the snipers before they have a chance to "set up". Thus by the time they are able to engage, they are already sufficiently weakened.

VIII. Vision Variable Analysis

The vision of a unit is an extremely important factor in determining it's effectiveness in battle. A high vision allows an agent to do many things that include making better decisions about where to move, attacking units that are further away, and avoiding dangerous units that are far away. The following is an analysis of homogenous armies of the same size with variable visions pitted against one another.

A. Infantry

In a battle between 40 infantry on each army, 1,000 simulations were run with varying visions for each army. The number of wins was recorded for each army. The following chart shows this data.

Max. Vis. of Army A	Max. Vis. of Army B	Effective Vis. of Army A	Effective Vis. of Army B	% Won by Army A	% Won by Army B
3	4	1.5	2	39.2%	60.8%
3	5	1.5	2.5	12.9%	87.1%
3	6	1.5	3	1.3%	98.7%
4	5	2	2.5	41.0%	59.0%
4	6	2	3	18.7%	81.3%
5	6	2.5	3	30.1%	69.9%

Infantry Vision Table

This table shows that as vision ratings are changed, battle outcomes are drastically different. It can be easily concluded that for units with relatively small vision modifiers, (units that cannot see very far) changing their maximum vision to be lower, decreases their effectiveness immensely. Another reason that infantry are effective heavily by their vision is the fact that they do relatively little damage, and have low hit points. This means that if a unit can see the infantry from a distance, it can hit the enemy before the enemy even gets within the range of the attacking unit. The conclusion is that vision is a very important factor for units with low damage and low hit points.

B. Mortar

In a battle between 40 mortars on each army, 1,000 simulations were run with varying visions for each army. The number of wins was recorded for each army. The following chart shows this data.

Max. Vis. of Army A	Max. Vis. of Army B	Effective Vis. of Army A	Effective Vis. of Army B	% Won by Army A	% Won by Army B
3	4	1.95	2.6	42.6%	57.4%
3	5	1.95	2.5	30.1%	69.9%
3	6	1.95	3	2.9%	97.1%
4	5	2.6	3.25	39.8%	60.2%
4	6	2.6	3.9	26.3%	73.7%
5	6	3.25	3.9	44.2%	55.8%

Mortar Vision Table

As expected, vision differentials have an impact on the outcome of battles between mortars. There is however, a caveat to this. Vision does not affect mortars as much as it affects infantry. The reason for this is most likely because of the splash damage that mortars can do. When mortars attack, they do a massive amount of damage to one target and a lot of splash damage to another target. This is usually not a function of vision since an attack will always result in a relatively large amount of damage. The reason for the big differentials in win % is due to the increased range of mortars with higher vision. In conclusion, it can be seen that vision *does* affect mortars, but not as much as infantry.

C. Sniper

In a battle between 40 snipers on each army, 1,000 simulations were run with varying visions for each army. The number of wins was recorded for each army. The following chart shows this data.

Max. Vis. of Army A	Max. Vis. of Army B	Effective Vis. of Army A	Effective Vis. of Army B	% Won by Army A	% Won by Army B
3	4	3	4	24.0%	76.0%
3	5	3	5	2.1%	97.9%
3	6	3	6	0.0%	100.0%
4	5	4	5	36.2%	63.8%
4	6	4	6	10.1%	89.9%
5	6	5	6	39.5%	60.5%

Sniper Vision Table

As expected, vision has a huge impact on how well snipers fare in battle. The major attributes that snipers rely on are their vision and their accuracy. If snipers cannot see far, they can often be picked off from a distance by other snipers, or they will be mauled in close combat. When a sniper's vision increases, its attack range increases as well. The above data shows that with a 3 to 5 vision differential, the army with less vision wins only 2.1% of the battles it engages in with the superior army. This is a direct result of the fact that the snipers with the better vision get *multiple* attempts to shoot enemy units before the enemies can even see the attackers. Due to the sniper's low hit points, the sniper must rely on eliminating opponents early, before they enter attacking range. This is a major reason that snipers are the agent most affected by vision.

D. Tank

In a battle between 40 tanks on each army, 1,000 simulations were run with varying visions for each army. The number of wins was recorded for each army. The following chart shows this data.

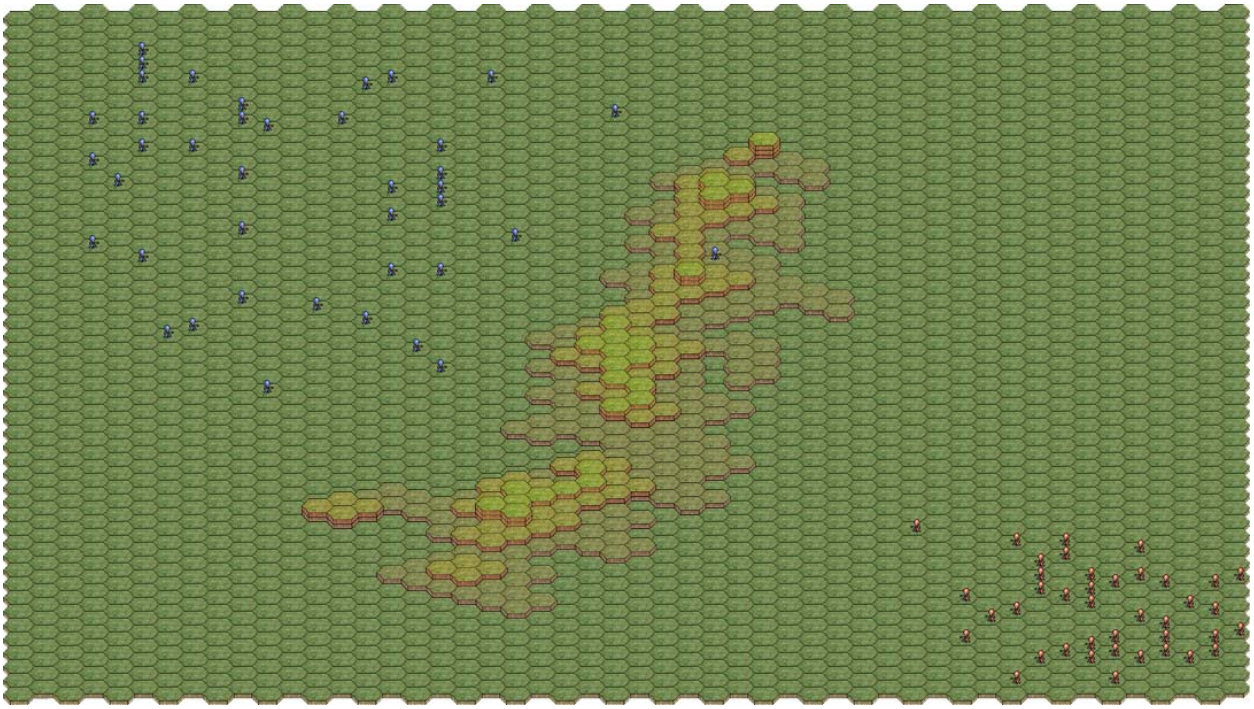
Max. Vis. of Army A	Max. Vis. Army B	Effective Vis. of Army A	Effective Vis. of Army B	% Won by Army A	% Won by Army B
3	4	2.4	3.2	38.6%	61.4%
3	5	2.4	4	21.2%	78.8%
3	6	2.4	4.8	14.3%	85.7%
4	5	3.2	4	41.0%	59.0%
4	6	3.2	4.8	32.1%	67.9%
5	6	4	4.8	38.3%	61.7%

Tank Vision Table

Though affected by vision, tanks are at a much smaller disadvantage if they are given a smaller vision. Whether or not a tank is given a large vision radius, the damage it deals, remains the same. The high amount of damage dealt by the tanks main attack and splash damage make the tank a dangerous unit regardless of its vision. The difference in win percentage here is largely due to vision giving a tank more information about attacks. A tank with a larger vision radius will be able to see more spaces to attack and therefore will be able to pick spaces that deal greater amounts of splash damage to enemies. For example, if there is a space that is a distance of 4 from a tank with a vision of 5 that will do high amounts of damage to the enemy, the tank can attack that space since it is in the vision of the tank. However, if that same scenario existed for a tank with a vision of 3, the tank would not be able to see this space and make an attack that would deal high amounts of damage. Instead, the tank would be relegated to choosing between fewer spaces that allow less damage to be dealt. The examples illustrates why vision is important to high damage units and low damage units alike.

IX. Spread Analysis

The sigma value of an army's initial distribution significantly affects its performance and apparent strategy in battle. After several runs of homogenous armies with different "spread" values, it was determined that various combinations of this variable in battle result in relatively predictable outcomes.



Blue Army with a Sigma of 10, Red with a Sigma of 3

A. Infantry

Sigma Army 1	Sigma Army 2	Army 1 Winning %	Army 2 Winning %
3	5	24.5%	75.5%
3	10	9.3%	90.7%
5	10	5.6%	94.4%

Infantry Spread Table

There is no question that infantry behave quite differently from mortar units with variation of spread values. The reason for the extremely high winning percentage of infantry armies with larger spread values than their opposition can be attributed to the infantry unit's speed. An infantry army is actually able to move fast enough to properly flank its enemy if it has a good enough difference in spread to do so. When an army successfully flanks its opposition, it is decidedly difficult for its enemy to succeed against the frequency of attacks it receives from a superior position.

The losses of the army with a better spread can be attributed to the chance that by random movement and positioning, it is unable to successfully create a flanking formation in time. This will usually result in the more spread-out army sending one to two units at a time into battle, where they are promptly slaughtered by the opposition. It is apparent that a more tightly distributed army is slightly more successful should such a slip-up occur, since they are able to quickly dispense with the trickle-in opposition using attacks from multiple units. A slightly more spread-out army will find it more difficult to take advantage of a flanking error from the opposition, since it will usually end up engaging its enemy in several smaller skirmishes.

B. Mortar

Sigma Army 1	Sigma Army 2	Army 1 Winning %	Army 2 Winning %
3	5	94.7%	5.3%
3	10	90.4%	9.6%
5	10	65.5%	34.5%

Mortar Spread Table

It would appear that in a battle comprised entirely of mortar units, it is far more beneficial to retain a tight formation, particularly against an army attacking with a relaxed arrangement. One could probably attribute this outcome to the devastating power of a mortar attack. When an army is positioned such that several mortars can simultaneously attack one single unit, it is certainly to that army's advantage. This is the scenario created when a more loosely arranged mortar army attempts to attack one that is more tightly knit. The former will usually have its units straggle one by one into the waiting attacks of several units of the opposing army, which practically eliminates most chance of a victory.

The variation in winning percentage, particularly what occurs when an army with a sigma of 10 is able to defeat an army with a sigma of 3, can be easily explained. If an army with a large spread is able to successfully flank a tighter army by chance of random movement, the flanking army will win every time. This is because surrounding an army with weapons that deal splash damage will result in several units being injured with every shot from the flanking army. The surrounded army will not be able to deal nearly as much splash damage to the more spread-out army, and will thus be eliminated quickly.

It appears that the effect of a tighter spread is far less relative than would have been originally predicted since an army with a sigma of 5 was only able to beat an army with a sigma of 10 65.5% of the time. The effects of spread values on a mortar battle thus appear to take on an exponential distribution, where the larger the numbers get, the less important spread becomes in outcome.

C. Sniper

Sigma Army 1	Sigma Army 2	Army 1 Winning %	Army 2 Winning %
3	5	26.3%	73.7%
3	10	39%	61%
5	10	29.2%	70.8%

Sniper Spread Table

The results of the sniper battles were rather surprising. It was originally hypothesized that due to the sniper unit's considerable range, that a tightly distributed army of snipers would not allow itself to be flanked by an opposing army. It appears though, that while the range of the sniper defrays some of the flanking behavior (the percentages for a small-sigma sniper army repelling an opposing army with a larger sigma are far better than those for infantry), snipers are still quite susceptible to this flanking strategy.

The jump in winning percentages for a sigma-3 army when it faces a sigma-10 army can be explained again by a sigma-3 army's ability to take advantage of flanking slip-ups. If the opposing army, by chance, happens to have its units deployed incorrectly to flank, it will result in its units trickling slowly into the main battle. A tightly distributed enemy will be able to deal with such an attack quickly. This is particularly true when it comes to sniper armies, since there is less chance of recovery from a flanking error, due to the units' long range.

D. Tank

Sigma Army 1	Sigma Army 2	Army 1 Winning %	Army 2 Winning %
3	5	29.4%	70.6%
3	10	37.5%	62.5%
5	10	28.8%	71.2%

Tank Spread Table

Interestingly, the outcome of the tank battle simulations was practically the same as the sniper simulations. Instead of rehashing the reasons that certain sigma values created different outcomes (flanking, taking advantage of errors, etc.), it is relevant to note some of the similarities between the tank and the sniper that make them behave similarly under different spread conditions. Both the tank and the sniper are slow units, and so actually positioning an army composed entirely of either unit into flanking formation is difficult to achieve before its enemy attacks. Both units also have long range, and a devastating attack, which aid both in the prevention of flanking *and* in maintenance of flanking formation once it is accomplished.

The major differences between the sniper and the tank lie in the hit-points of the respective units and the fact that that the tank's attack deals splash damage. It would appear that the differences are somewhat negated in a homogeneous battle, since the tank's heavy armor practically sheds splash damage. Also, since the damage to hit-point ratio for a tank is approximately the same as for a sniper, the two will survive practically the same number of shots from their enemy.

X. EINSTein Simulation Correlations

After some research, it was discovered that there was an agent based land-warfare model very similar to sWARm. The EINSTein model, or Enhanced ISAAC Neural Simulation Toolkit (where ISAAC stands for Irreducible Semi-Autonomous Adaptive Combat), uses many of the same rules that sWARm implements. In EINSTein, agents are randomly assigned a certain "personality," and will behave according to that personality. This can affect how often an agent is able to attack, how much it will move, and whether it will prefer attack or defense. This is very similar to sWARm's creation of various unit types, by which each unit type will behave very differently from another by nature.

Interestingly, movement selection in the EINSTein model is implemented practically the same way it is in sWARm. Each possible move is assigned a "penalty" based on the ratio of visible enemies to friends. The move with the lowest penalty value is what is chosen (in the case of a tie, a random move is selected). This is almost identical to the move scoring system used in the sWARm.

The actual execution of combat is where sWARm begins to differ significantly from EINSTein. In the EINSTein model, an agent is allowed to fire at *all* agents in its visible range in a given turn. When hit, an agent's status will move from "alive" to "injured", and then from "injured" to "dead". An injured agent will attempt to retreat from a battle until it is able to heal over a certain amount of time. A dead agent is, of course, removed from the simulation. This representation of damage is far different from sWARm's hit point and armor system. It was also decided that an individual agent ought to choose whether to fight or retreat based on the status of his army, rather than his own life status. The EINSTein model also doesn't take various types of

terrain or altitude into account in combat execution. Both of these are factors that can make or break the outcome of a battle in sWARm.

One incredible feature of the EINSTein model that sWARm was unable to implement is the ability to create "Meta-Rules." Meta-Rules alter the design of agent personality types and distribution to make for various controlled battle scenarios. For example, a certain Meta-Rule can prevent an agent from moving toward friendly agents once it is surrounded by a certain number of enemy agents (sacrificing itself to prevent bringing the rest of its team into danger). Another Meta-Rule could split an army into groups of agents that attempt to flank the enemy, or try to split the battle in many small skirmishes that are easier to win. It is feasible, given more time, that sWARm would be able to restructure the classes in the simulation to allow an average user to make such modifications to various unit types in a GUI interface, but currently, these medications are beyond the scope of sWARm.

sWARm was created without research about EINSTein because it was felt a post-creation comparison would be beneficial because similarities and differences could be highlighted and the reason for the similarities and differences could be analyzed. sWARm seeks to implement a larger breadth of attributes and variables than EINSTein, to compensate for sWARm's far less complex decision algorithms because the sWARm simulation designers do not have sufficient differential equations backgrounds to create EINSTEIN-esque differential algorithms.

XI. Conclusion

At the beginning of “project sWARm”, goals were lofty and expectations were high. By the end of the project, it was apparent that there are an infinite number of variables and variable interactions that can be utilized in a war simulation. sWARm utilized many features of battle simulations such as vision, battlefield altitudes and terrain, and differing unit types and strategies. While these are fundamental to any battle simulation, there is much more that can be added to make the simulation even more realistic.

Though sWARm does not even come close to mimicking real battle simulations, (it will be difficult for any simulation to do so) it does provide a solid foundation for simulations to build upon it. The logic functions and decision trees are self-contained, which means that decisions can be inserted into a decision tree to increase the complexity of the simulation. Additionally, each of the unit classes is given different, specialty behaviors which allow for the simulation to add more units with different abilities and different movement behaviors.

In conclusion, the complexity of sWARm allows it to be useful to those who wish to run simulations and compile test data and to those who wish to just run battles for fun and watch them play out in graphical mode. The sWARm simulation shows in many ways how the behavior of individual agents in an army is integral to the success or failure of the army itself. The sWARm simulation, though complex, is just a small sample of the complexities of battle simulations and agent based simulations.

XII. Simulation Execution Instructions

A. Create a new directory and unzip sWARm.zip to that directory.

B. Compilation instructions: (all commands are executed from the root project directory)

i. For batch mode:

```
mkdir batchclasses
javac –d batchclasses –sourcepath srcBatch –classpath batchclasses srcBatch/sWARm/Main/*.java
```

ii. For GUI mode:

```
mkdir GUIclasses
javac –d GUIclasses –sourcepath srcGUI –classpath GUIclasses srcGUI/sWARm/Main/*.java
```

iii. For 3D mode:

```
mkdir 3Dclasses
javac –d 3Dclasses –sourcepath src3D –classpath 3Dclasses src3D/sWARm/Testing/*.java
```

C. Execution instructions: (all commands are executed from the root project directory)

i. For batch mode

```
java –classpath batchclasses sWARm.Main.BatchHandler
```

Upon execution of BatchMode, a set of simulations will begin generating data in a text file in the root directory.

ii. For GUI mode

```
java –classpath GUIclasses sWARm.Main.GuiMain
```

Upon execution of GuiMain, a window will come up that allows parameter entry. To begin a simulation, enter up to 100 units per army, click “Load Parameters” and then “Start Simulation”.

iii. *For 3D Mode*

```
java –classpath 3Dclasses sWARm.Testing.SimTest
```

Upon execution of SimTest, a window will come up that allows parameter entry. To begin a simulation, enter up to 100 units per army, click “Load Parameters” and then “Start Simulation”. This will cause a full screen window to appear on which graphics are rendered.

Note: In 2D GUI mode the following colors represent the corresponding unit types: Infantry – Black, Mortar –Orange, Sniper – Green, Tank -White