Implementation of Real-Time Snow Layers in Game-Based Simulation

Dr. Michael D. Woodman
Bohemia Interactive Simulations
Orlando, Florida
michael.woodman@bisimulations.com

Peter Morrison
Bohemia Interactive Simulations
Orlando, Florida
peter.morrison@bisimulations.com

ABSTRACT

While many games have "snow" environments, they are only artists' representations. There is a requirement for game-based simulations to provide a realistic, real-time virtual snow environment for militaries that operate in snow. Among the many considerations for simulated snow are changing snow depth over time; depth of snow based on slope angle and direction relative to sun and wind; and varying snow depths on and around buildings, under trees, and on roads (which may be plowed). Because of these considerations, the snow layer is not uniform; it will require a tremendous amount of data, especially for large maps. Therefore, we decided to generate the snow procedurally, using optimized rendering. A simulation is not static; we have to consider the interactions: vehicles must have particle effects from driving through snow, they must leave tracks, they may sink into the snow, and they may plow through the snow. We need to calculate the force acting on each wheel for PhysX vehicles with a defined "floating zone" from the top of the snow layer which will depend on snow density. Of course, this also affects the snow height where the vehicle has driven, so we must update the height of the snow each time we simulate an object. The equation will take into account the mass of the object and the snow density. Similarly, soldiers are affected by the snow simulation; we must consider the increased difficulty of foot movement as well as the tracks left behind. This paper will discuss the tradeoff decisions, engineering solutions to creating snow layers, and lessons learned in developing snow layers for simulation. It will be of great interest to attendees who are considering implementing snow in their simulations, as it is important to understand the complexity of such a task.

ABOUT THE AUTHORS

Dr. Michael D. Woodman is the Senior Product Manager for Bohemia Interactive Simulations. Prior to joining BISim in 2013, Dr. Woodman was the Modeling and Simulation Officer for MAGTF Training Simulations Division at the USMC Training and Education Command. Before that, he served in various engineering and management positions in the modeling and simulation industry in Orlando, Florida, including two years with the USMC Program Manager for Training Systems. Dr. Woodman's last active duty assignment as a U.S. Marine Corps Major was as the USMC Project Manager for AV-8B and KC-130 flight simulators at the Naval Air Warfare Center, Training Systems Division. Dr. Woodman has a BA in Geography from the University of California at Los Angeles, MAs in Business Administration and Computer & Information Resources Management from Webster University, and a PhD in Industrial Engineering (Interactive Simulation) from the University of Central Florida.

Peter Morrison is the Co-CEO of Bohemia Interactive Simulations. Peter has been working for Bohemia Interactive since 2005. He studied Computer Science and Management at the Australian Defence Force Academy and graduated with First Class Honours in 2001. Peter had previously graduated from the Royal Military College, Duntroon, into the Royal Australian Signals Corp. Peter served as a Signals Corp Officer for several years, specializing in military simulation, and his final posting was to the Australian Defence Simulation Office (ADSO) as a Project Officer. Peter transferred to the Army Reserves in 2005 and joined Bohemia Interactive Australia as the

Lead Developer of VBS2, a position he held until he became Managing Director in 2007, and later CEO of the BISim Group. Peter is currently focused on bringing computer game technology to the military as an affordable alternative to traditional military simulation. He is also excited about the potential of this technology as a revolutionary training tool in other industries.

Implementation of Real-Time Snow Layers in Game-Based Simulation

Dr. Michael D. Woodman
Bohemia Interactive Simulations
Orlando, Florida
michael.woodman@bisimulations.com

Peter Morrison
Bohemia Interactive Simulations
Orlando, Florida
peter.morrison@bisimulations.com

"A Soldier trained in winter is also a good summer fighter; trained only in summer he is helpless in the winter!"
- Northern Warfare Training Center, Fort Wainwright, Alaska

While many games have "wintry" environments, they are typically only artists' representations. Computer games have been representing winter terrain for many years through the use of snow textures and custom 3D models. Recent examples include Skyrim®, a highly successful fantasy game by Bethesda, which includes a vast open world with extensive winter terrain, and Company of Heroes® 2, a real-time strategy game set on the eastern front in World War II. In both examples, the game worlds are developed from the ground up to give the appearance of wintry (typically snowy) conditions.

In The Elder Scrolls® V: Skyrim™, the snow is simply textured onto the terrain and 3D models (Figure 1). Snow has no depth, and it does not affect movement speed or any other aspect of the simulation. Additional effects such as particles of falling snow and condensation of breath are added by the game developers to increase immersion; however, snow does not affect the performance of in-game avatars in any way. There are, however, some interesting "mods" that have been created by third parties since the release of the game; for example, the Frostfall modification that adds aspects of winter simulation such as the ability to freeze to death if not wearing adequate clothing.

Figure 1. Winter scene in the computer game Skyrim™

Company of Heroes® 2 features both snow representation and simulation. Snow is textured on both the ground and 3D objects, but the snow layer on the ground can have depth that simulates avatars and vehicles "sinking" into the terrain, and snow can affect movement speed (Figure 2). Snow can provide either concealment or an obstacle depending on the scenario.

Figure 2. Soldiers advancing through the snow in Company of Heroes® 2

Although game-based simulations have been widely accepted as useful training aids, the requirement for snow to be simulated, in addition to being merely represented, is only just beginning to emerge. Game-based simulations such as Steel Beasts and Virtual Battlespace (VBS®) provide high-fidelity representations of terrain at the ground level and are used to train tactical decision making and other skills on a daily basis by many Western militaries. Given the possible significant impact of snow on movement, a realistic simulation of snow is important for many users especially those in the northern extremes.

Game environments have traditionally been developed from the ground up to represent the different seasons. For example, the Marnehuizen VBS terrain developed by the Dutch was built in four different variants to represent the four seasons (Figure 3). The summer version had grass and leafy trees, whereas the winter terrain had a white ground texture and trees with no leaves. This is the typical approach for both game-based simulation and traditional military image generators alike.

Figure 3. Marnehuizen VBS terrain, in summer, autumn and winter variants

The Swedish Defence Force recently published requirements for snow simulation for game-based training. The key requirement was to enable a given terrain database to be converted to support "snow layers," which should be configurable prior to the transformation and also support plowed roads.

This paper will describe the implementation of "snow layers" in VBS, and how it varied significantly from the "offline conversion" process originally envisaged by the Swedes. In later design meetings, the requirement was refined. Snow should affect line of sight, trafficability, and soldier and vehicle speed. Roads should be plowed; lakes should freeze over.

Among the many considerations for simulated snow are changing snow depth over time; depth of snow based on slope angle and direction relative to sun and wind; and varying snow depths on and around buildings, under trees, and on roads (which may be plowed).

The overall approach to simulating snow in VBS is as follows:

- Snow is simulated as a second height field layer to allow depth.
- Snow is generated procedurally (at run-time), not through an offline conversion.
- The generation process considers terrain objects (e.g. buildings) and features (e.g. rivers).
- Vehicle and avatar simulation was improved to deal with the effect of snow.
- Artificial intelligence (AI) was also improved to find appropriate paths through snow.
- Additional effects, such as particle effects, were added to increase sense of immersion.

We will examine the design analysis and tradeoff decisions that were made, the engineering solutions that made it a reality and lessons learned along the way that can help others who may desire to implement winter environments in their training systems.

REALIZING VIRTUAL SNOW: GEOMETRY, TEXTURE OR BOTH?

A key design decision was whether or not the snow layer should be represented as either a separate height field—an entirely new geometry layer—or as a texture that is modified and mapped on the regular height field, and the regular height field is raised where snow depth is required.

The texture method involves a snow texture that would be modified on-the-fly. It is faster at run time because no new terrain mesh is required, and it also forms nice smooth edges along linear terrain objects such as roads thanks to the natural utilization of bilinear filtering which all recent graphics processing units offer.

The advantage of the geometry method is that it is quicker to implement, and allows snow to be thickened very quickly. It also allows us to leverage our existing terrain level of detail algorithms to efficiently render thick snow on terrain out to long view distances.

The geometry method has been implemented but the key disadvantage is that it is difficult to generate smooth edges along roads. When defining the snow layer, it is possible to define the resolution of the terrain. High resolution means more condensed terrain vertices and smoother, more realistic-looking snow but there is a corresponding negative effect on performance. Lower resolution snow layers have jagged edges along roads. To reduce this "jagged edges" effect we implemented dynamic orientation of diagonals, which can be applied to any terrain layer (the underlying terrain or the snow). This reduces, but does not entirely eliminate, the problem. Figure 4 shows further differences between the geometry and texture approach. The image on the right uses the texture solution, and it is obvious that jagged edges along snow boundaries are removed entirely, without the need for dynamic diagonals.

Figure 4. Geometry (left) vs. texture (right) approach to the visual representation of snow.

Current work focuses on the texture approach. A procedural texture is being created that stores the snow values and bi-linear filtering produces the desired smooth edges. This approach is also very efficient because the resolution of the snow layer is no longer the driving factor to ensure the snow layer looks realistic.

PROCEDURAL TERRAIN GENERATION: THE SECRET INGREDIENT FOR HIGH-FIDELITY, MASSIVE TERRAIN

Early on in the design process we decided that VBS would support the generation of snow layers at run-time (procedural generation) rather than through an offline conversion process, as the Swedish requirement originally stipulated. Run-time generation is preferred because it uses less hard disk space (no need for multiple versions of a terrain); on-the-fly modification of snow depth, road plowing, frozen lakes, etc. is possible; the snow layer can react to objects created at run-time (e.g., biotopes, discussed shortly hereafter); and it allows snow to be simulated on maps not prepared for it initially. In other words, the procedural approach is backwards compatible. Snow can be simply applied to any VBS terrain as needed and at run-time.

The concept of *procedural* terrain generation is becoming more important in game-based simulation as terrain sizes become larger and terrain fidelity increases. The U.S. Army Games for Training (GFT) contract described a terrain size requirement of up to 1000km x 1000km, which could include millionsof trees and thousands of roads, creeks, and other linear features. It is not possible to include all such information in the terrain data itself because the physical size of the terrain database on the computer hard disk would become prohibitive.

Instead, a number of game-based simulations now generate terrain features at run-time based upon shape and material data embedded into the terrain database. Terrain features are generated using rule sets, so that the placement of trees, rocks, roads, and other objects is deterministic. VBS presently supports procedural biotopes (a region of habitat) and roads, with support for other linear features planned (e.g., creeks and power lines). These rule sets are defined in configuration files. The snow layer is considered another form of "procedural biotope" (Figure 5).

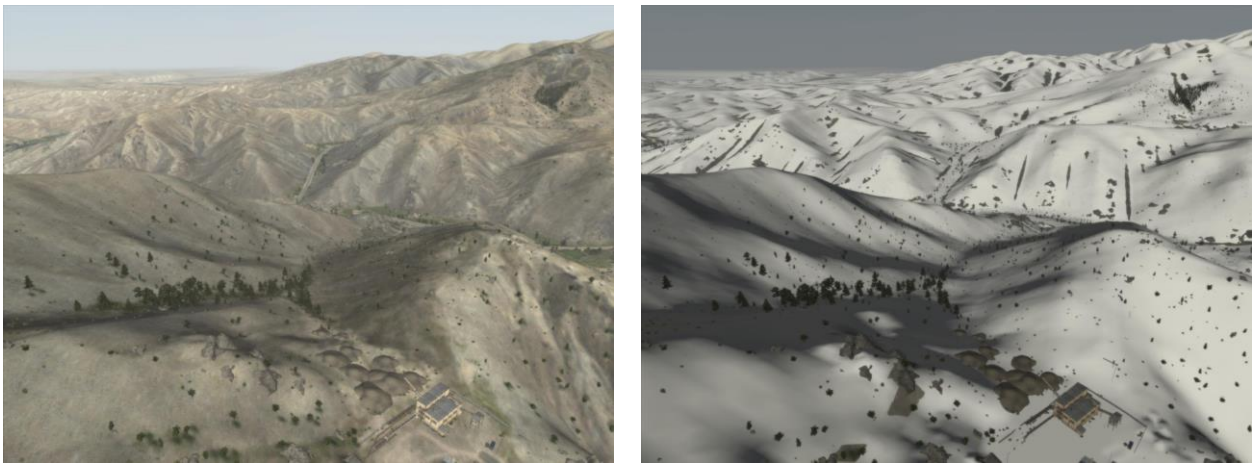

Figure 5. Snow "biotope" applied to a standard VBS terrain

The configuration of the snow layer is flexible, and example parameters include the following:

- As the snow layer is primarily represented through geometry (a second terrain height field), the initial height (snow depth) can be configured and then raised and lowered at run-time.
- Snow can be configured to appear only above or below certain altitudes, or both. This allows for the simulation of snow on mountains.
- Snow can be configured to appear only on terrain of a certain slope.
- Snow can be configured to appear a certain distance from roads and trees, simulating tree cover preventing snow fall and roads being plowed.
- The manner in which snow builds up around buildings and building entrances can be defined.

- Whether or not water will freeze, and the depth of water that will freeze, can be defined.
- Snow can be "rough" or "smooth."

The snow configuration is stored in an XML file that can be edited in a text editor or the VBS add-on application Silvador. Entirely new configurations can be loaded at run-time to immediately and dramatically change the appearance of the snow in a scenario.

Through procedural snow and biotopes, the old requirement to have multiple variants of a terrain to represent different seasons has been effectively removed. Through loading a new snow layer and biotope (for example containing deciduous trees without leaves, as opposed to deciduous trees with leaves), a terrain can be instantly turned from summer to winter without reloading the terrain itself or even the scenario.

DEALING WITH TERRAIN FEATURES: TO BEVEL OR CUT?

An important requirement for the dynamic snow layer is interacting with static terrain objects, which include buildings (both enterable and non-enterable), trees, and rocks.

We determined that there were two unique methods with which we could implement the snow around each building; one was to cut the snow layer at the walls of the building while the other was to bevel the snow around the periphery of the building.

Cutting technology was originally implemented to support avatars walking below the waterline on ships (i.e., the hull cutting through the water), and was extended to support underground objects. It was originally envisioned that the same technology would be used for buildings in snowy terrain, given that snow is represented as a second terrain layer.

However, the cutting technology turned out to be technically difficult and time consuming to implement. It also had the disadvantage, as can be seen in Figure 6 below, of being able to peek under the snow layer in certain circumstances. In this example, the prototype snow is about one foot deep. As shown, when looking through the window it is possible to see underneath the snow layer, which is obviously unacceptable.

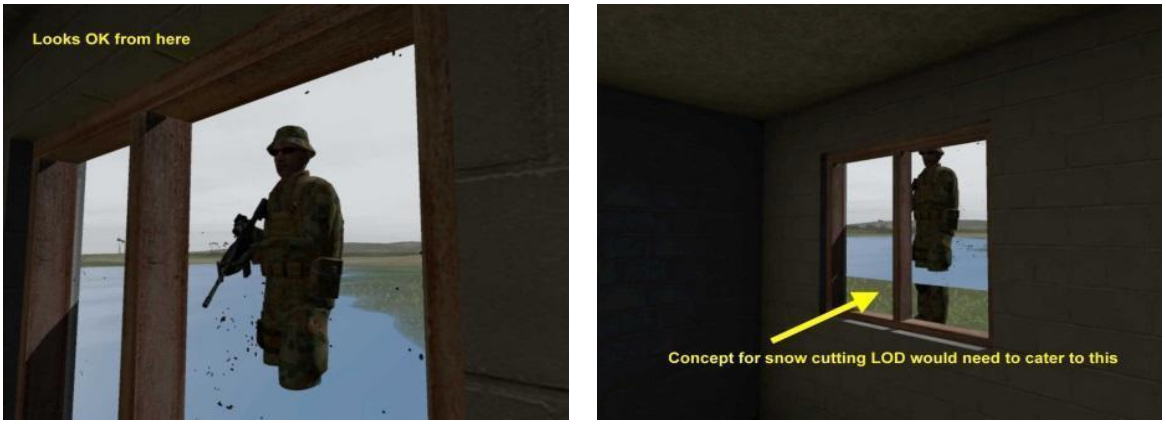

Figure 6. Example issue related to snow cutting around a building

Beveling the snow geometry was chosen as the preferred solution. It also increases realism: it gives the illusion of shoveled walks, trampled snow, and melting snow from the radiant heat of the buildings, as can be seen in Figure 7. Note that enterable buildings are treated by the simulation slightly differently. Snow is beveled around the entry point to a certain proximity, to allow avatars to still enter the building.

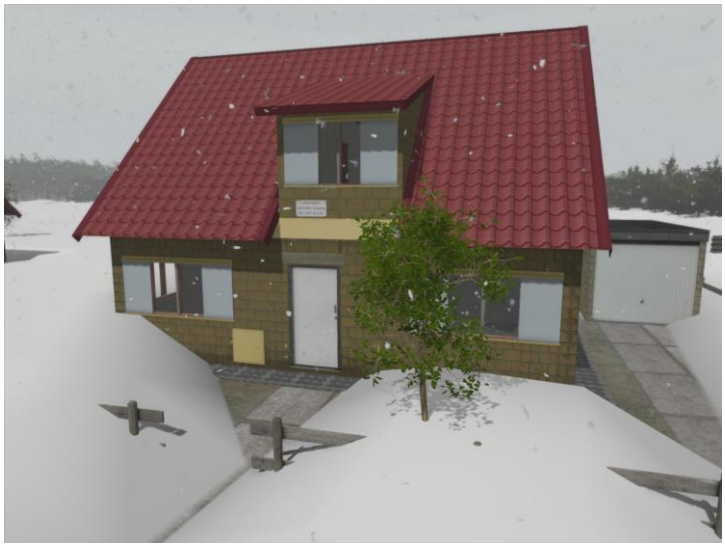

Figure 7. Beveled snow around a building, note entrances are cleared

Another challenge was to simulate snow accumulating on the roofs of buildings. As described in the following section, the snow layer is generated procedurally, but for performance reasons we cannot also procedurally create snow on the roofs of buildings simply based upon the existing 3D model data. This is because the process needs to examine every face of the model to confirm if it should be covered with snow, or not, and this is not efficient at run time (it has a complexity of O(n^2) where n is the number of faces).

We had to add an attribute to our building models to account for snow build-up on roofs. This is an automated, offline process based on the geometry level of detail within the model. Upward facing planes are considered so as to create an extruded snow cap. To be specific, 3D models need a "no_snow" selection in the geometry level of detail to exclude surfaces from generating snow caps. Shadows are correctly affected when the snow cap is present, but in the present implementation the snow cap does not affect AI line of sight or other aspects of the simulation. This may be addressed in future work.

There are many static objects besides buildings on the terrain that "interfere" with the smooth layering of snow, such as:

- rocks,
- fences,
- non-destructible debris and rubbish piles,
- berms, and
- trees.

We had to conduct an analysis similar to buildings. Should the snow be beveled around these objects or should the snow layer be cut? Cutting was determined to be appropriate for the following reasons:

- These objects do not have doors or windows, so the likelihood of "peeking" under the snow layer is extremely remote.
- There would be no human interaction to clear entrances or paths around these objects.
- The objects are not heated internally as a building would be.

For trees that protrude through the snow, we simulate thinner snow around the tree (as in the real world) by checking proximity to the tree when the snow layer is generated.

While it has been argued that the sun can warm objects, particularly large rocks, causing radiant heat to melt the snow around them, that does take time (and sunshine, of course!) and most of the objects on the terrain would not

have this effect on the snow layer. We therefore decided to simply allow these objects to be buried in the snow, with the appropriate distortion of the snow layer if the object is large enough.

We also implemented a simulation of freezing water, and frozen lakes and rivers would be passable by in-game entities. Ice is very different from snow but had to be considered in our analysis. In the real world, lakes and rivers can freeze over without the presence of snow. Just as likely, it can snow without lakes and rivers being frozen. We decided that there is no training value in watching lakes and rivers freeze over, so our implementation includes lakes and rivers that are frozen solid and trafficable. Whether or not water freezes over is defined in the snow layer definition itself. Of course, the amount of snow on top of a lake or river has an effect on the trafficability, and snow provides better traction than ice.

SNOW SIMULATION: MORE THAN JUST A PRETTY LAYER

VBS simulates a range of moving models (entities) including both vehicles and human characters (avatars), and the Swedish requirement stipulated that thick snow should have an impact on entity movement. Snow should also provide concealment by affecting line of sight.

For simulation purposes, all entities know about the snow height field. The height of the snow above true ground level and the density of the snow are the most important aspects of the simulation. Combined with the weight and surface area of the entity (variable in the case of the avatar, depending on posture – standing, kneeling, or prone), the simulation determines how deep (if at all) the entity will sink into the snow.

We had to create new functions within the simulation to account for snow height and density. For example, new functions return the original snow height as it was generated and also snow height after compression by either vehicles or explosions. The ratio between the two determines the density of the snow and, therefore, influences how far entities penetrate the snow and hence their ability to move on the surface.

In essence, we implemented the idea of modified buoyancy to simulate the interaction of rigid body objects with snow. Both avatars and vehicles "float" in the snow depending on their mass and the snow density.

There is a simplified approach to vehicle movement through the snow based upon vehicle power and size. Heavy tracked vehicles such as tanks will sink but move fairly well, while smaller wheeled vehicles will have much greater difficulty. Presently, no vehicle can get completely stuck in the snow. They will just move very slowly, because we want to avoid situations where AI becomes completely stuck. Future work will examine a more realistic friction-based model similar to the current VBS simulation of boggy ground.

Snow compression was implemented for vehicle movement and explosions. For vehicles, a snow change event is generated every two meters. Snow is bevelled in accordance with the vehicle position and distance between the wheels, with the lowest point being in the middle of the cut (directly under the vehicle) and then raising roughly diagonally out to the regular height of the snow. The ratio for the snow bevel is calculated as mass divided by radius, meaning that the bigger the mass and the smaller the radius, the more compression occurs.

For explosions, a similar approach is implemented. The radius of the compression corresponds to the indirectHitRange of the explosion (a parameter defined in the VBS configuration files on a per-ammunition basis), with the maximum depth defined as half of this parameter.

Clearly, the compression algorithms are relatively simple and they will be improved in future. Current challenges include making the effect still look realistic if the snow height field is sparse, and making the system efficient; it is not technically feasible to change the snow layer every simulation frame for every vehicle in the scenario.

Regarding snow density, there is a threshold defining the minimum mass needed to compress the snow further at the current density and this threshold rises as the density increases. The more mass an object has, the more snow gets compressed each simulation sub-step until it will be so compressed that it cannot be compressed any more with the given mass.

While avatars are slowed down by deep snow, the simulation does not yet directly affect their levels of fatigue. The VBS fatigue system uses modified Pandolf equations to track total metabolic energy cost of movement which is converted to biomechanical power in watts. This power and thus work is broken up into two aerobic pathways, one anaerobic pathway, an acidosis buffer, respiratory rate buffer, and muscle integrity/central nervous system power buffer. Each buffer or pathway has different characteristics of depletion and recovery and is utilized at varying levels of physical activity. The integration of movement through snow with the fatigue model is considered future work. Even though avatars and vehicles sink into the snow, both footprints and vehicle tracks are visible on top of the terrain as would be expected.

INTELLIGENT PATH-PLANNING: SIMULATING THE LACK OF SNOW SHOES

The VBS path-planning system involves both strategic and operative path-planning. The well known A* algorithm is implemented in VBS by default, which relies on a "cost map" to determine the shortest and fastest path between two points. Terrain is divided by a homogeneous grid of square tiles, called LandGrids. The default size of each is 40x40m.

At the strategic level, the A* algorithm steps forward from the LandGrid at the start position to the LandGrid at the end position and evaluates the cost of all possible routes. The "cost" depends on features on the LandGrid and the type of entity, e.g. vehicles as opposed to avatars. Roads might have less of a cost than off-road, and hence roads might be preferred depending on the entity type. Once the A* process is complete, we have a final path represented as a list of LandGrids.

Operative path-planning starts when strategic path-planning ends and involves searching smaller fields along the strategic path. The operative grid squares are 2.5m in size (16 per LandGrid). Various optimizations such as an operative cache are used to ensure that the entire operative grid does not need to be loaded into memory. The result of the operative path-planning is an array of positions (nodes) that moving models will follow as a path through the LandGrid.

The "cost map" is modified by the snow layer depending on the depth of snow and routes through the snow. Snow that has been compressed by a vehicle will be a more appealing route for avatars than heading on a straight line through deep snow. Likewise, plowed roads will be much more appealing for vehicles than deep snow. In Figure 8, the cost map is visible, with darker squares indicating increased cost (i.e., deep snow). The tank has previously compressed the snow to create a path, which the avatar AI is preferring for increased speed.

Objects that might have been impassable previously will become passable if covered in snow. Water surfaces will become passable for entities after freezing, and at the same time impassable for ships.

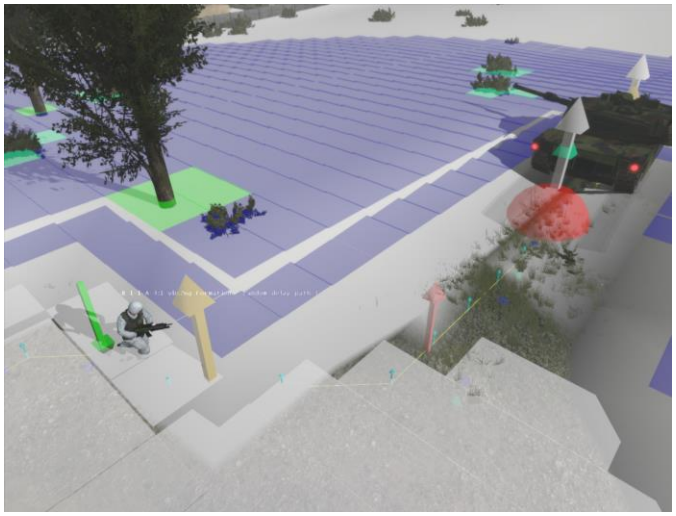

Figure 8. AI "cost map" showing high cost to move through compressed snow (blue shading)

INCREASING IMMERSION: ADDING THAT COLD, WINTRY FEEL

In addition to the snow layer itself, numerous effects have been added to make the wintry environment more immersive. Work was undertaken to properly simulate fog, which adds significant ambiance to the winter environment. The new fog simulation consists of two separate effects: *extinction*, which is the effect of light being out-scattered as it passes from the originating object to the viewer, turning the object progressively darker as the viewer moves farther away, and ambient light being in-scattered into the trajectory of the light along its way. The two effects combined cause far objects to lose contrast and fade towards a uniform color. The user also has significant control over the new fog simulation, including the amount of fog, the decay rate, and the base altitude. This means it is easy to have fog lie in valleys in a realistic manner, whereas previous fog was a global concept covering an entire terrain at all altitudes.

A new particle effect to simulate falling snow has been added (Figure 9). Various parameters can be changed to simulate the effect of wind and also represent either heavy or light snow fall. A new snow particle effect was created for vehicles, similar to the dust created by vehicles when travelling over dusty terrain. A plowing effect is visible when vehicles drive through the snow.

Figure 9. Example snow fall

FUTURE WORK AND LESSONS LEARNED: INCREASING THE DEPTH OF SNOW SIMULATION

The current implementation of snow layering in VBS is considered the minimum necessary for tactical training in a game-based simulation. Beyond considerable visual fidelity – falling snow, snow on surfaces, and fog – the presence of snow impacts the behavior of entities in the form of movement speeds, path planning and visibility (situational awareness). Further, the snow is impacted by the behavior of entities – becoming compacted as said entities pass over the snow. All this is achieved in a computationally tractable manner.

Future work is focused on optimizations and improving the visual fidelity and simulation of the snow:

- The implementation of bi-linear filtering will smooth out the edges of snow along roads and other linear features. Similarly, the smoothing of snow tracking behind vehicles will improve the visual appearance.

- The simulation of thermal imaging does not presently take snow into account. The thermal model needs to be updated accordingly.

- Snow is presently controlled through the biotope definition and script commands. Biotopes presently apply to an entire terrain but they need to support localized implementation. The mission editors need new user interface elements to give administrators finer control over the snow and frozen water (for example, specifying which lakes are frozen and which are not).

- Snow is not automatically applied to vegetation, and hence a new biotope with snow-covered trees needs to be manually loaded (separate from loading the snow layer). We are building additional biotopes that will be provided with future versions of the software; for example, four deciduous forest variants for the four seasons with appropriate 3D tree and underbrush models.

- Vehicle tracks and footprints currently disappear after a period of time, when in reality this will only occur once new snow falls.

- The snow shader needs to be improved to include subsurface scattering and texture to represent small irregularities in the snow surface.

- Ice thickness needs to be simulated to allow breaking the surface and sinking into the water. Presently, a frozen river or lake is frozen from the surface right down to the underlying ground.

- Both the simulation of vehicles being bogged in snow and avatars becoming fatigued as they move through snow need to be improved.

The development of snow layers in VBS has been an interesting project with many lessons learned, including:

- Procedural generation of the snow layer at run time was determined to be the most effective and flexible implementation.

- A procedural snow texture, with dynamic raising and lowering of the underlying ground layer to represent snow depth, appears to be the most appropriate implementation of virtual snow (instead of a second layer of geometry), although work is ongoing.

- Determining snow covered surfaces within buildings and other 3D models at run time proved to be inefficient, and additional information had to be added to all VBS objects to support snow caps.

- Having vehicles compress snow is a nice feature but performance is a significant issue depending on the number of vehicles in a scenario and the complexity of the snow layer.