Simulating Urban Traffic in Support of the Joint Urban Operations Experiment

Dan Speicher, Deborah Wilbert
Lockheed Martin
Simulation, Training & Support
Advanced Simulation Center
Burlington, MA
dan.speicher@lmco.com, dwilbert@lads.is.lmco.com

ABSTRACT

Providing background vehicles to serve as sensor clutter is an important component for experiments involving sensor detection of clandestine opposing forces. The ClutterSim has been used since 1998 to cheaply simulate large amounts of civilian vehicles for the purpose of confusing or overloading sensors. Clutter vehicles need to be simple enough to be cheap to simulate, but realistic enough that OPFOR vehicles are difficult to detect when they are performing their tasks.

In support of the Urban Resolve 2004 (JUO) experiment, it was necessary to further increase the intelligence of the clutter vehicles to mimic realistic urban traffic patterns, including traveling on the correct side of the road, stopping at traffic lights, parallel parking, stopping for pedestrians, and so forth. This was perhaps the most challenging requirement in the history of clutter development, since it conflicted with one of the primary methods for keeping clutter cheap to simulate: clutter vehicles are completely unaware of all other vehicles, including other clutter vehicles. A further complication was that any solution had to work properly over the network, since clutter is commonly simulated on multiple machines.

This paper describes the solution to this problem: the creation of an network-synchronized traffic intersection controller which permits appropriate behavior of clutter vehicles at an intersection while preserving clutter vehicles' ignorance of the presence of other vehicles and only adding a minimal amount of network traffic.

ABOUT THE AUTHORS

Dan Speicher is the lead developer of the ClutterSim application. He is a Software Engineer at Lockheed Martin Simulation Training & Support Advanced Simulation Center (LM STS-ASC) in Burlington MA. Dan holds a B.S. in Electrical Engineering from Cornell University.

Deborah Wilbert is a researcher and developer in distributed simulations and the author of a study on ClutterSim functionality. She is a Staff Software Engineer at Lockheed Martin Simulation Training & Support Advanced Simulation Center (LM STS-ASC) in Burlington MA. Deborah holds a B.S. in Computer Science from the Massachusetts Institute of Technology.

Simulating Urban Traffic in Support of the Joint Urban Operations Experiment

Dan Speicher, Deborah Wilbert
Lockheed Martin
Simulation, Training & Support
Advanced Simulation Center
Burlington, MA
dan.speicher@lmco.com, dwilbert@lads.is.lmco.com

INTRODUCTION

In sensor systems, clutter is anything that induces a sensor response similar to the response elicited by the presence of an intended target. Clutter can be caused by terrain features, man-made objects, natural phenomena, or background traffic. Each of these could, and should, be modeled in military simulations.

From a high level view, the "sensor system" also includes automated target recognition (ATR) algorithms and human participant intellect. For some joint military operations, such as time critical mobile target detection and urban situational awareness acquisition, background traffic clutter is of special importance. This is because traffic clutter not only offers signature snapshots that can confuse individual sensors, but its behavior over time can camouflage enemy vehicles from intelligent algorithms or analysts.

Therefore, behavior is the aspect of background traffic that is especially important to simulate in these sorts of joint level experiments. However, simulated behaviors come at a high cost.

Traditional Computer Generated Forces (CGF) systems such as Joint Semi-Automated Forces (JSAF) can and have been used to simulate limited numbers of high fidelity clutter vehicles. But this approach quickly becomes prohibitive on the theater level scale needed to support joint experiments. JSAF continues to be used to provide selected high value clutter, but cannot be used to provide the bulk of background traffic.

The computational cost of realistically simulating behavior is a well-known problem. One approach used historically to limit the cost of modeling a behavior is to simulate it at an aggregate level. However, this approach does not provide adequate behavioral camouflage when the operational task is to locate individual enemy platforms.

In addition to the computational cost, state changes resulting from behavior models must be transmitted over the network in distributed simulation. The inclusion of long-range sensors and high-level views in joint experiments prohibit the use of some simple short-cut strategies to reduce network bandwidth requirements (such as divide-and-conquer).

This paper describes a specialized application developed to meet the critical requirement of adequately modeling urban traffic clutter for joint experimentation while minimizing computational and network costs. While that application's original very simple behaviors were well suited for joint time critical target detection experiments, dramatic enhancements were needed in order to adequately support the joint urban operations experiments which have taken on so much importance in today's military conflicts. This paper details those necessary enhancements.

BACKGROUND

History

In 1998, the Joint Advanced Warfighting Program (JAWP) and the J9 directorate of the US Joint Forces Command (JFCOM) began to investigate how experimentation can be used to discover new concepts for future joint military operations (Torpey, Wilbert, Helfinstine, & Civinskas, 2001). The first joint experiment, J9901, postulated that in the 2015 timeframe it could be possible to track enemy movements well enough to successfully neutralize mobile enemy Theater Ballistic Missiles (TBM). The

scenario included a hypothetical network of multi-modal radar sensors mounted on satellites and unmanned air vehicles together with unmanned ground sensors. An attack operations cell used these sensor assets along with ATR and human analysis to identify mobile targets in a hostile country. Without background traffic clutter, this task would pose no difficulty for the operations cell.

Therefore, a realistic J9901 scenario required the inclusion of about ten thousand background clutter vehicles. Although JSAF was available, the STOW 97 experiment showed it was only capable of providing 18 vehicles per (200 MHz P6) simulation computer (Ceranowicz, et al, 1999).

The ClutterSim application was developed to provide the necessary background traffic clutter at an acceptable cost. The ClutterSim was founded on the JSAF code base. However, the vast bulk of JSAF functionality was stripped away, leaving only what was needed to provide simplistic behaviors for clutter traffic. Simple road movement, damage calculation, response to sensor footprints, and publishing entity state were the major existing JSAF functionalities that were reused by clutter vehicle models. Additionally, tick-based scheduling was replaced with event-based in order to conserve more processing.

The resulting ClutterSim was capable of supporting a thousand clutter vehicles per simulation computer. Ten stations running ClutterSim provided the required 10,000 clutter vehicles in the actual J9901 experiment. Test runs showed that up to double that number could be supported, given proportionally more computers to run on (Ceranowicz, et al, 1999).

The ClutterSim application has evolved throughout the history of J9 joint experimentation, culminating in the current version used to support Joint Urban Operations (JUO). The operational task of detecting and tracking enemy forces in an urban environment where those forces are trying to blend with the civilian population requires significantly more sophisticated background traffic clutter.

The background traffic clutter must behave believably both individually and in aggregate like an urban population. Individually, vehicles and pedestrians must stay in their lanes (sidewalks), obey traffic signals (usually), and park at appropriate locations. In aggregate, the background clutter must simulate the normal ebb and flow of urban traffic, which reflects the activities of commuting, dining, shopping, and sleeping. ClutterSim was extended to support modeling of all these phenomena.

This added modeling is conducted in a rich terrain environment including a high density of buildings, roads, and other features. The added burden of the new functionality and more complex environment reduced the number of clutter vehicles able to be supported per station to several hundred. However, optimization efforts were successful in bringing that number back up over a thousand, and the current JUO scenario incorporates approximately 100,000 clutter entities (Ceranowicz & Torpey, 2004).

DESIGN PHILOSOPHY

More is Better

In the domain of simulation, a common goal is to model the real world as realistically as possible. However, a critical characteristic of real world background traffic that must be modeled realistically is quantity. As the saying goes, "Quantity has a quality all its own."

In order to support the required quantity of background vehicles, some sacrifice in the accuracy of individual vehicle models is both necessary and acceptable.

Compute-intensive vehicle dynamics and the physics of interaction with terrain skin are very important aspects of manned vehicle trainers for tank crews. But they are essentially irrelevant for modeling background clutter. Ignoring them in clutter vehicles introduces no discrepancies that the players can perceive.

Similarly, expensive "intelligent" automated behaviors need not reside in individual vehicles, as long as background traffic behaves believably en masse. ClutterSim allows the user to easily provide simple guidance to all clutter vehicles, forgoing the necessity of having each clutter vehicle figure out what to do on its own.

These sorts of shortcuts, along with careful optimizations, allow ClutterSim to provide the quantity of background vehicles needed to achieve a scenario that is realistic overall.

Ignorance is Bliss

In order to achieve quantity, it is imperative that vehicles spend little or no time "thinking" during the simulation.

Normal JSAF vehicles spend most of their processing time examining the terrain and vehicles around them in order to plan or react. Conversely, clutter vehicles are designed to be as ignorant as possible of the world around them while still retaining the capability to act believably.

ClutterSim conducts route planning in an initial step, by computing routes based on the terrain's road topology. Once a route is planned, the clutter vehicles no longer need to access the terrain for any purpose except elevation lookups to clamp the vehicle to the terrain when it moves.

Clutter vehicles also completely ignore other vehicles. Normally a simulator does not even subscribe to receive information about other vehicles from the network.

This lack of awareness does come at a price. If a clutter vehicle behavior must model interaction with other vehicles or terrain elements, it is difficult to write and the resulting code tends to be complex. However, in ClutterSim, these behaviors are the exception, so the huge gains in performance are well worth that price.

Keep it Simple

Since the purpose of ClutterSim is to provide a multitude of vehicles, it must be relatively simple for users to create and provide guidance to clutter vehicles, else the task would be overwhelming. In particular, the user should not need to create or direct vehicles individually.

In order to create a scenario, the user need only specify the number of clutter vehicles to create, the types of vehicles, the type of behavior the vehicles should follow, and the area in which to create the clutter. Optionally, the scenario may also contain "forbidden zones" which clutter vehicles must avoid, and timing constraints, which affect traffic flow. These features are described in more detail later.

The normal way of creating a ClutterSim scenario file for initialization is via the standard JSAF GUI. A JSAF editor allows the user to input the desired values and draw the area to create the clutter on the map. This template can then be saved and used to create the actual clutter vehicles with a click of a button.

Alternatively, the user can completely automate this process if he has the necessary information already available in one of three supported formats. A JSAF Persistent Object scenario, DIS logger file, or formatted text file can be used as input to existing conversion utilities for creating ClutterSim templates.

Furthermore, the library defining the file format for the clutter templates provides a simple API. Thus, it would be easy for a programmer to create a utility to import a new file format.

Users also have a limited control of clutter vehicles while an active scenario is being simulated. Using a normal JSAF GUI, users can dynamically add forbidden zones to a scenario or task clutter vehicles directly.

Clutter vehicles can be selected individually, or groups can be selected using the "rubber band" GUI idiom. The selected clutter vehicle(s) can then be given a direct order. They can be destroyed, halted, or given a new route to follow.

To keep processing overhead low, ClutterSim does not perform terrain analysis for routes that are provided as overrides during running scenarios. The user is responsible for providing routes that already take terrain into account. The blissfully ignorant clutter vehicles will follow their new routes exactly. If the user orders ground clutter vehicles to cross a deep body of water, they will miraculously do so.

Despite this limitation, it is easy for users to compensate for the lack of intelligence in individual clutter vehicle behavior models, when necessary, to maintain the required level of realism for background traffic.

Share and Share Alike

As described earlier, ClutterSim was based on JSAF code. This allows the ClutterSim to take advantage of the many man-years of development and testing that have gone into JSAF.

Not only was code reused, but it continues to be shared between JSAF and ClutterSim in an ongoing basis. An advantage of sharing code over simple reuse is that enhancements developed for either application are automatically available in the other. For example, whenever new types of vehicles, control mechanisms, or user interface features are added, they can be used by either application without doubling the development.

A less obvious advantage of code sharing is that it provides the best possible camouflage for JSAF vehicles to conceal themselves among clutter vehicles.

If vehicles attempting to conceal themselves among clutter used different route planning or road movement algorithms, they would be too easy to pick out. When changes were made to these modules to support urban clutter traffic, enemy JSAF vehicles inherited the same code, allowing them to merge in seamlessly.

CLUTTERSIM FEATURES

Types of Supported Vehicles

All vehicle types supported by JSAF are also supported by ClutterSim. However, ClutterSim only uses a small subset of the information that is needed to configure a vehicle type for use in JSAF. In general, ClutterSim only relies upon knowledge of a vehicle's physical size, domain (land, sea, or air), and maximum speed. More information can be provided to ClutterSim, such as supplies or munitions but these currently do not affect clutter vehicle behavior.

ClutterSim provides different levels of support for vehicles in different domains, as explained in the following subsections.

Ground Vehicles

When ClutterSim was originally created in 1999, only ground vehicles were supported. Today, most functionality is still built around ground vehicles, and most uses for clutter require ground vehicles.

Forbidden zones, traffic rules, and user intervention are only supported by ground vehicles.

Pedestrians

Lifeforms, a subtype of ground "vehicle", are treated specially by ClutterSim. For current purposes, all Lifeforms are considered pedestrians, even when the specification represents a dismounted infantry.

In ClutterSim behavior, pedestrians walk on sidewalks (unless they are crossing a road). Since terrain databases currently usually omit information regarding sidewalks, pedestrian clutter assume they always exist by the side of each road.

This assumption could lead to visual anomalies if a building or body of water were placed adjacent to the edge of a road. However, so far these potential problems have not undermined the utility of pedestrian clutter for experiments conducted in urban terrain.

Air and Sea Vehicles

ClutterSim supports air and sea vehicles in a limited fashion. Any type of air or sea vehicle supported by JSAF can be created, but their behaviors are simplistic even by ClutterSim standards.

When created, air vehicles will pick a random route over the terrain to follow. Similarly, sea vehicles will pick a random route on available ocean to follow.

Neither sea nor air vehicles act very realistically. While enhancing air and sea clutter vehicle behavior would not be difficult, there has not yet been any demand for more sophisticated air or sea clutter behaviors.

Classes of Clutter Behavior

When the user specifies an area in which to create clutter vehicles, that area is distinguished by the class of clutter that will be created. For historic reasons, the area is called a "Clutter Control Point" (CCP) and the type of control point determines the class of behavior of the clutter vehicles created for that CCP. The following sections describe those behavior classes and how each act.

Static Clutter

Static clutter vehicles are the simplest form of clutter supported by ClutterSim, and the least expensive to simulate. When created, they randomly choose a spot along a road, parallel park there, and never relocate.

The parallel parking functionality is part of the ClutterSim implementation of traffic rules. Static clutter vehicles use the same system as other clutter vehicles, which prevents two or more clutter vehicles from attempting to park in the same location.

Mobile Clutter

Mobile clutter was the first type of clutter behavior supported in original version of ClutterSim implemented in 1999. Although mobile clutter has been upgraded to use the traffic controls recently introduced to support the JUO program, mobile clutter exists almost unchanged from its original form.

When created, a mobile clutter vehicle will pick two random endpoint locations on the road network within its designated CCP, create a route between these two points, then drive continuously along that route, parking for a while and then reversing direction whenever an endpoint is reached.

While mobile clutter was effective for its original purpose of providing background clutter, an experienced user could usually determine whether or not a given vehicle was clutter by observing its perceived route based on its track history.

Source/Sink Clutter

The Source/Sink Clutter behavior class was created to simulate refugees fleeing from a specified area. The user specifies one or more source and sink CCPs. Over time, at a rate controlled by the user, clutter vehicles will be created within a source CCP, take a route to a randomly selected location inside a sink CCP, parallel park there for a few minutes, then vanish.

The clutter vehicle is then recreated inside a randomly selected source CCP. The process repeats indefinitely, thereby creating a steady stream of simulated refugee clutter traffic.

Crowd Clutter

Crowd Clutter is a new class of clutter behavior added for Joint Urban Operations and is currently only partially implemented.

Crowd Clutter is intended to simulate crowds or mobs of people and allow the ClutterSim operator limited control over their behavior. When a Crowd Clutter CCP is created, the specified number of pedestrian clutter will be created at random buildings within the area. This pedestrian clutter will then travel from their buildings or origin to form a mob of people in the center of the CCP.

The resulting crowd can be dispersed by the operator on demand. In that case, the pedestrians return to the buildings from which they came then vanish. Alternatively the operator can direct the mob move to a new specified location.

While this class of behavior models the appearance of crowds adequately for the first phase of the JUO experiment, it does not currently block traffic in the simulation. The next step planned in the implementation is to use ClutterSim forbidden zone functionality (described in a subsequent section) in order to cause crowd clutter to block roads.

Commuter Clutter

Commuter Clutter is another new class of clutter behavior added for Joint Urban Operations. In urban contexts, Commuter Clutter is now the most important class of clutter behavior supported.

One of the requirements for the JUO program was to simulate realistic city traffic patterns. Traffic is more active at certain times of the day (i.e. rush hour), with observable patterns occurring in the traffic flow.

One solution considered was to enhance the existing mobile clutter behavior to be more configurable. In this scheme, the user could vary the rate at which mobile clutter would be generated, depending upon the time of day. Popular and unpopular destinations might be modeled by specifying some sort of attractor and repulsor locations.

Together, these extensions might have enabled mobile clutter to adequately represent the gestalt appearance of urban traffic flow. However, the utility of the resulting mobile clutter would still be vulnerable to analysts identifying Clutter vehicles by observing their randomly wandering routes over time.

Therefore, the concept of commuter clutter was developed, made possible by the recent addition of building types to terrain databases. New versions of terrain database identify buildings as being one of about 130 different types, such as restaurants, gas stations, houses, office buildings, and so forth (Miller, et al, 2003). Access to building type information provides a better basis for realistic appearing simulated behavior than an amorphous concept of attractors and repulsors.

Before creating commuter clutter, the user must first design a commuter profile. A commuter profile is a list of destinations that a person would travel to throughout a typical day, including the amount of time that they would want to spend at each destination and rules for how to transition from destination to destination.

The concept of the commuter profile is most clearly illustrated by an example. Part of the profile for an office worker might look like this:

- At 0700, +/- 30 minutes, travel from home (building of type apartment or residence) to work (building of type office building).
- 30% of the time, at 1130, +/- 15 minutes, travel from work to lunch (building of type restaurant).
- 45 minutes, +/- 10 minutes after arriving at lunch, return to work.

- At 1630, +/- 45 minutes leave work and go home. 20% chance of stopping at a gas station on the way home. 50% chance of stopping at a grocery store for an hour on the way home.

- At 1900, +/- 30 minutes, there is a 20% chance of going out to a movie for 2-3 hours.

Within a profile, a building can be characterized as being "unique", meaning that each separate instance of a Clutter vehicle following this profile will remember what building it picked as its destination and use that same building for future trips to that building type. For example, this allows a commuter to travel to the same office every day, but patronize a variety of different restaurants for lunch. Destination buildings can be any subset of the building types defined in the terrain database, or a set of unique buildings designated by the user. Search distances may also be specified, which allow a profile to indicate how close the next destination building must be to the current location in order to be considered for selection.

Events in a profile depend on the day of the week as well as the time of day, which allows a profile to specify different events on a day-by-day basis. A profile can also specify a percentage chance that the clutter vehicle will park inside the building (modeling a parking garage within the building) rather than search for a nearby spot to parallel park on the street and then spawn a clutter pedestrian to actually walk to the target building.

Using the new commuter behavior class, it is fairly easy to develop a set of profiles to model heavy rush hour traffic with lighter traffic at other times. The resulting traffic flows in a believable fashion, with the bulk of vehicles shifting from residential to industrial and commercial areas and back again at appropriate times.

Forbidden Zones

Forbidden zones give users indirect control over all clutter vehicles' behavior simultaneously. They represent areas that clutter vehicles should avoid for experiment-specific reasons.

Forbidden zones are represented by arbitrary areas on the terrain. They can be specified either at initialization time or during an experiment. Users can draw forbidden zones using the JSAF GUI by creating arbitrary polygons on the map.

If a forbidden zone is added during an active experiment, existing clutter vehicles outside the zone will replan their routes if they intersect the new zone.

The user may choose which action each clutter vehicle should take if it is within a forbidden zone when it is created:

- Ignore the forbidden zone; Used, for example, if road blocks are set up to deny access to an area, but vehicles already inside the area are allowed to leave at their own pace.

- Flee the forbidden zone; Used, for example, if a firefight begins in an area, giving clutter vehicles a good reason to flee the area as quickly as possible.

- Destroy self; Used, for example, if a minefield is created with enough density to assume clutter vehicles trapped in the area would be destroyed.

- Delete self; Used, for example, if a bridge is completely destroyed, to remove any clutter that might otherwise be left hanging in mid-air.

Traffic Rules

Simulated vehicles in an urban setting must follow a basic set of traffic laws in order to behave realistically. Furthermore, traffic laws were established for a reason and a set of well-known conventions must be followed by individual urban vehicles (simulated or not) in order to accommodate high traffic density without many collisions.

Clutter vehicles stay on their own side of the road. The appropriate side, left or right, can be configured to accommodate different conventions in different countries.

Clutter vehicles pull out of the travel lane when they parallel park. Due to lack of source data in the terrain databases, it cannot be cheaply determined whether there is sufficient room off the side of the road to parallel park, so currently all roads are assumed to have a wide enough shoulder or a separate lane for parking. Parking information is distributed over the network so that all clutter vehicles will know which parking spots are currently available.

ClutterSim supports modeling stop signs at intersections. Stop signs may be placed for all incoming directions or all directions except one, on which traffic is unrestricted.

Traffic lights are also supported, but with limitations. Terrain source data is not available for determining traffic controls at intersections. Instead, a simple algorithmic approach is used to assign traffic controls to each intersection. While the selected assignments may result in believable traffic behavior, they are unlikely to allow optimal traffic flow. Additionally, in order to keep the light cycles for an intersection in sync across all machines, all lights run on a light cycle of the same length.

In the simulated ClutterSim traffic system, with non-optimal traffic controls and inflexible pre-planned vehicle routes, gridlock becomes a serious threat. The danger is especially acute in urban areas with short distances between intersections. It is easy to envision a scenario in which lines of vehicles backup into nearby intersections, forming a loop of no-go traffic. This can become the kernel of unbreakable gridlock that will rapidly cascade throughout the entire scenario.

While much the same thing could happen in real life, traffic levels in actual urban areas are self-adjusting. If traffic densities reach gridlock levels, fewer people choose to commute in cars. Additionally, in real life cars involved in a gridlock situation would seek alternate routes and disobey traffic rules in order to get out of the gridlocked area.

In order to cheaply prevent an unrealistic gridlock scenario in the simulation, ClutterSim implements a failsafe mechanism. Clutter vehicles that get stuck in traffic for too long are placed in an invisible limbo state until the road that they need to reach is no longer blocked.

More realistic solutions to the gridlock problem are being explored in ClutterSim, including allowing vehicles to use the shoulder to drive around blockages and forcing vehicles to replan their routes if they become blocked by gridlock.

Once the traffic rules described in this section were added to ClutterSim, background urban traffic behaved in a much more believable fashion. However, the primary purpose of ClutterSim is to provide background movements capable of concealing the location and identity of enemy vehicles. Although clutter vehicles behaved more believably, their behavior diverged from that of the enemy JSAF vehicles, making them easier to identify.

However, since the two applications share their code base by design, it was relatively easy to provide the JSAF vehicles with access to ClutterSim's traffic intersection logic. This enhancement provides JSAF vehicles with a behavior to conduct road movement in the exact same manner as clutter vehicles, thus re-establishing their effective camouflage.

NETWORKED TRAFFIC MODEL

The Problem

As discussed earlier, providing an adequate quantity of background traffic requires that clutter vehicles be ignorant of all other vehicles (including other clutter vehicles). This means that individual clutter vehicles cannot try to avoid collisions with other vehicles, which results in behavioral anomalies. Additionally, when two clutter vehicles do happen to be at the same location at the same time, they cannot detect each other so they pass through each other. This results in visual anomalies. In an urban environment, high vehicle densities and increasingly sophisticated sensors and tracking algorithms must be supported. Therefore, in order to provide believable sensor clutter, the anomalous collisions must be addressed.

Potential Issues

Processing Overload

The traditional CGF solution would be to add awareness and intelligence to the clutter vehicles. The vehicles would watch out for potential collisions and adjust their own speed or path to avoid them. However, this solution isn't practical for clutter vehicles. Clutter vehicles would no longer have any performance advantage over normal CGF vehicles.

Information Overload

Clutter vehicles are ordinarily hosted on several machines, so avoiding collisions between arbitrary pairs of clutter vehicles requires a distributed solution. Clutter vehicles, like CGF vehicles, publish entity state updates containing data that could be used to avoid collisions. However, clutter simulators would have difficulty processing the sheer volume of information available in entity state updates at the rate they are produced as the scenario approaches the scale of a joint experiment.

While the JSAF system does process entity state updates, only the data from a small subset of the participating vehicles is generally received. The rest of the entity state traffic is filtered out geographically. This is possible because the relatively small number of CGF vehicles simulated by each JSAF tend to remain

clustered near each other (on the simulated terrain). The large number of clutter vehicles simulated by a ClutterSim do not tend to cluster together, so the heuristic of geographical filtering would not reduce the volume of necessary entity state data significantly.

Imprecision

Urban traffic must be modeled very precisely in order to maintain travel in lanes and prevent collisions. While arbitrary precision could be achieved between clutter vehicles being simulated on the same physical machine, precision is an issue between vehicles simulated by separate simulators. Network latency and the use of dead reckoning as well as dropped packets (in a "best effort" transmittal scheme) introduce unacceptable imprecision for preventing collisions in tightly packed traffic.

One possible solution for this would be to adopt a scheme in which one simulator would simulate all the clutter vehicles located in the same geographic area. Then, clutter vehicles in danger of colliding with each other would be simulated by the same machine, and they would have local access to accurate position information.

One difficulty with this scheme is that it would require implementing a clutter vehicle migration mechanism, which would be complex and add significant overhead.

A more significant problem is that it would likely be impossible to assign geographic areas to different simulators so that loads are balanced fairly throughout the experiment. Certain features, such as bridges, act as bottlenecks at certain times of day. At those times, the simulators responsible for bottleneck locations will be unfairly overloaded and overwhelmed if the load is reasonably balanced at other times.

The Solution

The solution is to employ a replicated, networked traffic model. The clutter vehicles can remain blissfully ignorant of other vehicles, and need not engage in explicit CPU-intensive modeling of collision-avoidance, sensors, and behavior. External coordination makes clutter vehicles appear to be intelligent enough to avoid collisions.

By replicating the model in each simulator, the positional accuracy of co-located clutter vehicles is essentially achieved and the distribution of information specific to traffic control is much more efficient than deriving the necessary information from entity state updates.

Each simulator incorporates a traffic controller for each intersection on the road network. Whenever a clutter vehicle is about to travel on a road, it must register with its traffic controller for the next intersection on that road and follow commands from the traffic controller on whether or not the vehicle is permitted to enter as well as when and how fast it is allowed to travel. The traffic controller has enough knowledge of each vehicle heading towards the intersection to prevent collisions between them.

The traffic controller's task is simplified since all vehicles are following the established set of traffic laws: travel on the correct side of the road, obey red lights and stop signs, observe standard right of way rules, etc. Additional constraints on movement are also assumed, such as traveling at a constant speed until slowing for a stop.

Given this set of assumptions, the traffic controller can analyze the four different potential collision situations between vehicles deterministically. A deterministic algorithm allows the intersection controllers to be replicated within each simulator.

Rather than rely upon entity state data, the traffic controllers in each simulator send and receive the registration information each time a vehicle enters a road or passes through an intersection.

All simulators receive enough traffic information in the registration messages to treat remote clutter vehicles as if they were their own local vehicles. Each traffic controller does exactly that, and simulates the state change of remote vehicles locally. This allows traffic to flow smoothly through intersections regardless of which simulators are actually simulating the vehicles involved and eliminates the imprecision that would be introduced by latency in receiving entity state updates.

Registration messages are smaller than entity state updates and occur less frequently, since the replicated traffic control algorithms automatically account for some state changes. The resulting volume of incoming traffic is only about 10% of the corresponding entity state information.

The following sections describe the four potential collision situations and how each is handled by the traffic controllers.

Entering Roads

Whenever a vehicle intends to enter a road segment at an endpoint, the traffic controller must determine when there is a sufficient amount of room between the

entering vehicle and the preceding vehicle. Given that all vehicles entering from that endpoint have registered their speed and entry time, it is easy for the controller to time the entry of the new vehicle.

If a vehicle is turning onto a road from an off road location, such as a driveway or a parking spot, the traffic controller must not only ensure adequate space behind any predecessors, but also prevent the new vehicle from cutting off any approaching vehicles. Even assuming a right hand turn, this calculation is more expensive since it involves vehicles on either side of the entry point and the cutoff margin is larger. Additionally, the controller has to do extra math to figure out exactly where vehicles are located, since the calculation is not based on time, as it is for entering the road from the previous intersection. Fortunately, this case is relatively infrequent compared to entering the road from the endpoint.

Overtaking Vehicles

Once a vehicle is on a road, the traffic controller must ensure that the new vehicle will not overrun the vehicle in front of it. The easiest way would be to prevent the vehicle from moving any faster then the vehicle in front of it, should one exist. However, the traffic controller allows the new vehicle some leeway, just not so much that it would overrun the preceding vehicle before the next intersection.

When registering, the entering vehicle specifies its desired speed. The traffic controller indicates the maximum allowed speed when it is granted permission to enter a road.

Traffic Backups

When a vehicle enters a road, it is only granted permission to travel the full distance to the next intersection if there are no vehicles preceding it. Otherwise, the sum of the lengths of all the vehicles currently preceding it (plus some cushioning between each) is subtracted from the allowable distance. As predecessors successfully travel through the next intersection, the allowed travel distance is increased appropriately. Thus each vehicle will ultimately stop at the right place in line, should there be a backup of vehicles waiting to enter the upcoming intersection.

In order to perform this calculation, each vehicle indicates its length to the traffic controller when registering.

Entering Intersections

Each traffic controller has all the information it needs to determine when each vehicle heading towards its intersection will arrive. By using the travel distance update mechanism, the traffic controller has complete authority over which vehicles are actually allowed into the intersection and when. The traffic controller uses this authority to simulate real world traffic control mechanisms, such as a stop lights or four-way stop signs, and to pace the entrance of vehicles such that collisions are avoided.

Remaining Concerns

The first implementation of this solution worked surprisingly well, but did fail under certain conditions.

Deadlocking

The synchronized clock used by the traffic controller accounts for latency between machines, but it is still possible for it to have small errors due to fluctuations in the network or dropped packets. Therefore, it is possible that two simulators would try to add a local vehicle to the same road at nearly the same time, and for both of the simulators to think that the other simulators owns the lead vehicle. This would result in a deadlock condition, where each simulator will not allow its own vehicle to proceed until the supposed lead vehicle has progressed onto the next road segment.

To break this deadlock, each traffic controller in each simulator locally models the granting of permission for remotely simulated vehicles to progress. If all simulators were perfectly synchronized, then the correct lead vehicle would move on to the next road segment on all simulators (both local and remote) at the same time.

In the case of the potential deadlock, the sequencing of which lead vehicle progresses first will be reversed between the simulators involved. However, in this case, the update messages from the local traffic controllers will resynchronize the state of the clutter vehicles after the vehicles involved pass through the intersection.

Latency

Even if the clocks used by the simulators are synchronized perfectly, network latency introduces a potential race condition. Two simulators might both add a vehicle to the same road in the same position.

While this is not as severe as a deadlock condition, it leads to the anomaly of two vehicles traveling on top of each other.

When the race condition is detected, the vehicle that should be trailing is granted less travel distance than the intended front vehicle. The anomalous condition then corrects itself before the vehicles reach the end of the road.

JSAF CGF Limitations

Before adding the traffic controller, JSAF CGF vehicles could act almost identically to clutter vehicles by using the same movement behavior. The addition of the traffic controller adds a new set of constraints on JSAF CGF vehicle behavior if they are to participate in the collision avoidance scheme and be well camouflaged by clutter vehicles. Specifically, in order for the rather complex timing interactions between the Clutter movement model and the intersection controller to actually work properly, standard JSAF vehicles must disable their physics-based hull models before they can take advantage of the Clutter traffic controllers. In scenarios such as JUO, that is an acceptable compromise and the JSAF CGF vehicles can behave in exactly the same fashion as the Clutter vehicles.

CONCLUSION

Simulating background traffic is a critical requirement in joint experiments. Recent emphasis on urban operations posed new challenges to model high-density urban background traffic with adequate realism.

Some of these needs were met with the addition of the Commuter Clutter and the incorporation of travel rules into the basic road movement model. However, successfully coordinating clutter vehicles to avoid frequent collisions in high-density urban conditions required the introduction of the innovative redundantly-simulated networked traffic model described in this paper.

This solution appears to work well in JUO experiments, providing sufficiently high-density background traffic with adequate realism. There are many ways in which the realism of the urban clutter vehicles could be improved further, but low-cost innovative approaches are needed since the emphasis is to provide large numbers of vehicles.

REFERENCES

Ceranowicz, A., & Torpey, M. 2004. Adapting to Urban Environments. *Interservice / Industry Training, Simulation & Education Conference (I/ITSEC).*

Ceranowicz, A., Torpey, M., Helfenstine, B., Bakeman, D., McCarthy, J., Messerschmidt, L., et al. Fall 1999. J9901, Federation Development for Joint Experimentation. *Simulation Interoperability Workshop, 99F-SIW-120.*

Miller, D., Cauble, K., Bakeman, D, Torpey, M., Helfinstine, B., Ceranowicz, A. 2003. Extensions to the CTDB Format to Support Joint Experimentation. *2003 Spring Simulation InteroperabilityWorkshop, 03S-SIW-133.*

Torpey M., Wilbert D., Helfinstine, B., & Civinskas W. March 2001. Experiences and Lessons Learned Using RTI-NG in a Large-Scale, Platform-Level Federation". *Simulation Interoperability Workshop, 01S-SIW-046.*