Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

A MOTION ENVIRONMENT FOR WIRELESS COMMUNICATIONS SYSTEMS SIMULATIONS

Nathan J. Smith
Trefor J. Delve

Communications Research Laboratory
Motorola Labs
1301 East Algonquin Road
Schaumburg, IL 60196, U.S.A.

ABSTRACT

We describe the environment and motion systems used in a parallel, discrete event large scale wireless simulator. The simulator is capable of supporting user motion on multiple environment types (different types of streets, buildings etc.) and provides a unified and intuitive interface to users whilst being efficient for the systems that make use of it. This is achieved by making use of a hierarchical environment description. With this approach, users can provide different levels of detail as required, whilst the motion systems have a simple interface to interrogate the environment. As there is a close coupling between the environment and the RF data required by the wireless simulator (which is considerable in size), this too is represented in a hierarchical manner. This allows a more efficient use of system memory with only the data that is required being loaded.

1 INTRODUCTION

As radio communications systems grow to be increasing complex it becomes more desirable to use simulations to model these systems. Simulation modeling allows a researcher to gain a more controlled understanding of the system's operation. However, as simulations are just a model of actual systems, the accuracy of results gathered from simulations is limited by how closely their model represents reality. The more detailed the model, the more accurate the results can be. Detail, though, comes at a price.

Radio network simulations are notoriously slow and can use a frightening amount of computing resources. As a result, researchers find themselves working in a minimalistic environment enabling only those simulation features that are deemed absolutely necessary. One feature that is often ignored in wireless systems simulations is user motion. User motion, though, is the core of most modern wireless systems.

While motion had an arguably minute effect on the relatively simple wireless systems of the recent past, its affect on the operation of modern and future wireless systems can be quite profound. Simulated user motion will become increasingly important as system designers begin to tackle modern features such as high-bandwidth wireless applications.

This paper will address the need for simulated user motion by describing a motion environment designed for use with discrete event wireless communications systems simulations. It will first describe the simulation architecture in which the motion system has been designed. The physical environment description will then be detailed. Lastly, an overview of the implemented motion and physics system is included.

2 SIMULATION ARCHITECTURE

The core of the simulation architecture is DaSSF, a parallel discrete event simulation framework. As the primary focus of the simulator in which we are implementing this motion environment is RF interference calculations, the simulator architecture has been designed with these calculations in mind. Each simulation consists of a number of entities connected together through event routers.

Event routers are entities that route events between simulation entities. By connecting entities through event routers, we are able to limit the number of inter-entity connections. All events being sent from one entity to another are sent indirectly through an event router. The event router receives the event and determines how to route it based on the event's destination address. In principle, event routers function in much the same way as network routers.

Each router and all of its connected entities are grouped together into a logical process. The logical processes are spread across all available processors. Each router has a number of mobile and basestation entities as well as an RF entity attached to it. The RF entity is responsible for periodically calculating the received signal interference of all RF entities that are attached to its router. The frequency of this update depends on the simulated system's technology.

While the partitioning scheme that has been chosen for this simulation architecture is extremely efficient for RF calculations, it is not necessarily the most efficient partitioning for distributed motion simulation. It does, though, provide a framework that lends itself well to both the environment and motion solution that we present in this paper. A more in depth discussion of this architecture can be found in the paper, Use of DaSSF in a Scalable Multiprocessor Wireless Simulation Architecture (Delve and Smith 2001).

3 ENVIRONMENT

The basis of any motion system is the environment in which simulated users move. For a wireless system, that environment can vary dramatically in size. Some simulated systems may be as small as a few adjoining rooms while others may incorporate entire cities. The environment portion of our motion system must be capable of representing this broad spectrum of environments. It also needs to work under varying computing environments including shared-memory and distributed-memory parallel execution. Lastly, the environment needs to efficiently handle our RF description, thus enabling us to scale not only the size of the environment, but also handle the increasing amounts of RF data associated with that environment.

How, though, do we efficiently represent physical systems of varying degrees of complexity? Our solution lay in the path of a hierarchy.

3.1 The Environment as a Hierarchy

Looking at the world around us, we see that objects can be classified into a hierarchical relationship with each other. Almost every object can be said to be contained by exactly one other object. From the perspective of an RF system simulation, all physical objects in the simulation, such as buildings, streets, and trees, would be contained by the bounds of the simulation space.

For example, a simulation of the RF systems in Chicago could be broken down such that Chicago is the bounds of the simulation space. Chicago would then contain many buildings, one of which is the Sears Tower. Likewise, the Sears Tower could contain many floors, one of which is the Observation Deck. This relationship is depicted in the following Figure 1.

It is not difficult to see how other environments could be partitioned and built using such a system. Before the environment hierarchy is fully usable, though, there are a number of things that must be described so that we can give form to the nodes in the hierarchy. Also, a general set of rules that can be applied to all of the nodes in the tree is helpful in defining a consistent environment.

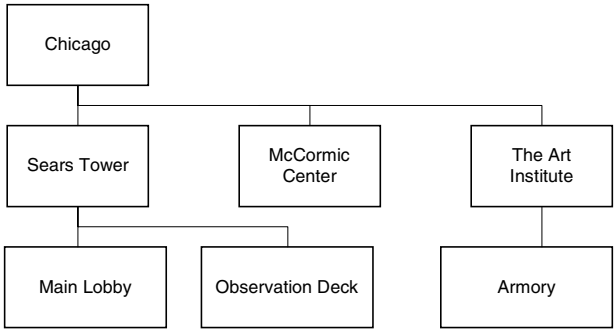

Figure 1: Hierarchy of Buildings and Floors in Chicago

3.1.1 General Rules

In the environment description, as in life, there are certain limitations that are imposed by the world. As we are creating a model of a physical environment, we would like for simulated users to be able to move through the environment in understandable and predictable ways. We would also like to be able to control how those users move through the simulated world. One step towards this control is access restrictions.

3.1.1.1 Access Restrictions

It is likely that we will want to prevent pedestrians from walking on expressways and likewise prevent vehicles from driving around on building floors. However, without some form of access restrictions, our users are free to roam throughout the environment. Thus, each node in the environment hierarchy has an associated list of allowed and restricted users and user types. This provides us with two keys on which to discriminate.

As a user attempts to access a node in the environment, it will be discriminated against based on its user type. If its user type is accepted on that particular node, it will then be checked to ensure that its user id in particular has not been listed on the disallowed list. If either check fails, the user should not enter the restricted environment node.

Additionally, each node can have a base range of speeds associated with it. Any user that moves into an environment node must be capable of moving within the base range of speeds which that node has published. Again, this prevents pedestrians from entering expressways as pedestrians are unable to move in the range of speeds required for entry.

In Figure 2, three users are trying to access the park. User 1 will be rejected because it is of type vehicular which is restricted in parks. User 2, on the other hand, has an acceptable type, type cyclist, although it will still be rejected because its range of speeds does not intersect with the allowed range of speeds in the park. Only User 3 will be allowed entrance to the park because not only is it of an acceptable type, type pedestrian, its range of speeds also intersects with the allowed range specified by the park node.

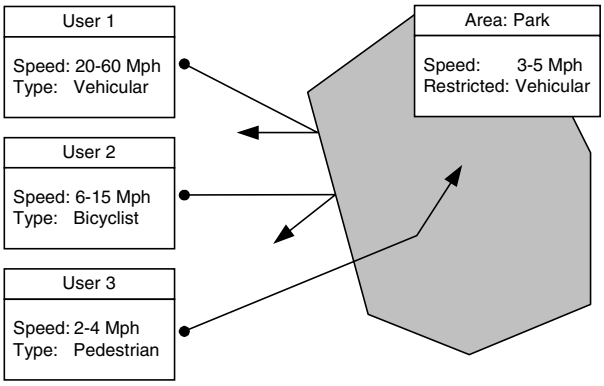

Figure 2: User Type Access Restrictions

While in principle it would be relatively simple to implement a system that checked users as they were entering and leaving environment nodes, due to the distributed nature of our physics engine (see Section 4.5) our implementation relies on the honor system. We leave it up to each user to determine its fitness for entry into each node in the environment.

3.1.1.2 Layout Restrictions

There are two primary restrictions on the layout of objects in the environment: objects on the same plane cannot partially overlap each other and objects must be wholly contained within the bounds of their parent.

The restriction on areas stems as a requirement for access restrictions and path finding. If two areas partially overlapped each other (and one was not contained by the other), it would create differing descriptions for the same simulation space potentially allowing restricted users access to an otherwise inaccessible area. The same reasoning supports the restriction that child objects must be wholly contained by their parent. If an edge of a child were allowed to spill past the edge of its parent, it would create a situation of ambiguity.

Keep these basic restrictions in mind as we discuss the primitives out of which an environment can be built.

3.1.2 Vector Streets

Vector streets are mathematical vectors that connect exactly two points in our physical environment. These vectors, like all nodes in our hierarchy, have various associated attributes. These attributes include speed and direction.

The speed attribute indicates not only the range of speeds at which users may travel, but also allows for the definition of varying ranges of speed depending on user type. For instance, we could have a vector that represents a street and its sidewalks. On such a street we may want to force all vehicular traffic to move at speeds between 30 and 45 Mph. However, because that vector also represents the sidewalks along that street, we wouldn't want to force our pedestrians to move at a brisk 30 Mph walk. By assigning the same vector two conditional speed ranges, the condition being the type of user that is moving on the vector, we are able to differentiate between these two types of users and allow our pedestrians to move at a more comfortable three miles-per-hour.

The same can be said for direction. While there may be one-way streets that apply to cars in a city, it is unusual to limit street-side pedestrian traffic to a single direction of flow. By making the direction also conditional on the type of user, we allow ourselves flexibility in how we assign attributes to the vector streets.

3.1.3 Polygonal Areas

Polygonal areas describe areas of arbitrary shape and size. They are used to represent everything from the floor of a building to the surface of entire cities. The environment description supports any valid single contoured convex or concave polygon that exists on a single plane. Complex polygons, such as those shown in Figure 3, are not supported for the simple reason that we are modeling reality, and you will not find bow-tie buildings or parks that twist over on themselves in real life.

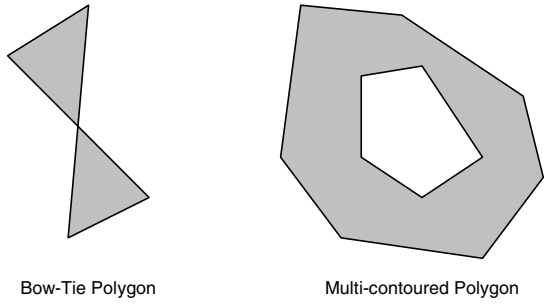

Figure 3: Illegal Polygon Shapes

A useful feature of polygon areas is that they may contain other objects in the hierarchy. This would allow you to create a polygon that seemingly had two contours. The second contour would simply be another polygon nested inside of the first.

Polygons are described as a series of edges. Each edge can have associated attributes that apply to only that edge. Additional user restrictions, for example, are a common addition on a polygon's edge. These additional restriction attributes further limit what types of users are allowed to cross an edge. This limits users as they not only enter, but also exit, a polygonal area.

Edge-by-edge user restriction application is useful when creating areas such as garages. While it may be acceptable for vehicles to enter through the garage door, you may want pedestrians to use a side entrance. This can be accomplished by allowing only vehicles access through the

garage door edge of the polygon thus preventing pedestrians from crossing that polygon edge. Similarly, pedestrians would be the only user types allowed access through the side entrance. The remaining polygon edges would be set so that no access is allowed through their edges. This would prevent users from walking or driving through the walls of a building.

3.2 Example Definition

Using only the primitives that we have described, vector streets and areas, we are now able to describe an environment in which our simulated users can traverse. Figure 4 below depicts an environment consisting of a building, shown as a semi-transparent shell of three floors, and two pathways, shown as a pair of gray lines on the third floor of the building.

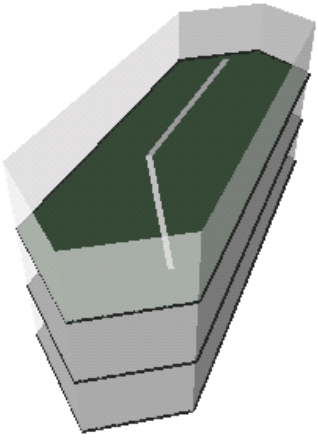

Figure 4: Sample Motion Environment

As the following Figure 5 depicts, the building is built as a tree of areas. The pathways are then contained within the building.

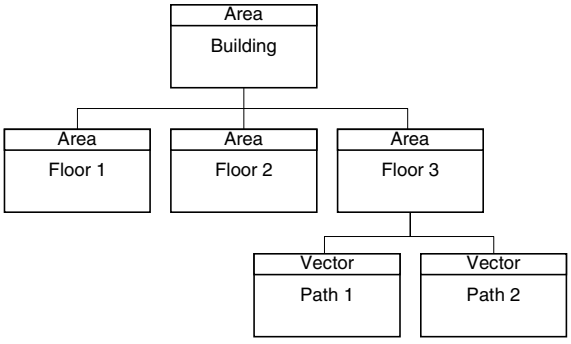

Figure 5: Hierarchy of the Sample Motion Environment

3.3 Simulation Interface

A popular division scheme used in large environment systems as of late is to partition the environment at physical boundaries, placing only a small portion of the environment on each processing node being used in the simulation. This can provide a desirable load-distributing mechanism. Our simulation, though, cannot be efficiently divided using geographical boundaries. Instead, as described in Section 2, it is divided and spread based on an even distribution of RF entities. This has a direct impact on the distributed aspects of the environment.

Because there is no physical division of the environment into partitioned pieces, each processing node used to run the simulation must have a complete copy of the entire simulated environment. While this is not terribly efficient in terms of memory usage, it does save in communications between processing nodes. In order to further limit the amount of overhead caused by the environment on the rest of the simulation, objects in the environment are forcibly static. Had we chosen to implement doors that open and close or operating traffic lights, a mechanism would have been required to update each copy of the environment with these changes.

Probably the most important aspect of the environment system interface is how the user interacts with the environment hierarchy. Iterators into the environment tree are sent to all users at the beginning of the simulation. These iterators have the same interface as that of an STL iterator and allow the users to traverse the tree in two dimensions: ascending and descending through parent-child relationships as well as moving laterally to nodes sharing a common parent. The iterator into the environment provides a useful and intuitive interface into what could otherwise have been a complicated data structure with which to work.

3.4 Environment Scalability

Despite the limitations on physical partitioning schemes that are placed on the environment due to its design, the fundamental choices that have been made allow for scalability.

3.4.1 Arbitrarily Sized Environments

As previously noted, a key requirement of our environment description is the scalability of environment complexity. The hierarchy provides a mechanism of scalability that we may not have had with other descriptions.

In the description of a simulated system in Section 3.1, we chose Chicago as the root of our environment tree. Chicago, we described, contains many buildings each of which contains many floors. However, we would not have to describe the environment as such. If we were interested in only outdoor environments we could quite easily remove the insides of the buildings from the description. The structure of the environment is not fundamentally changed as we still have a tree of objects in our simulation. Likewise, we could expand our system boundaries to include all of Illinois or all of the United States.

The scaling of detail also works in the opposite direction. We could bring the scale of the environment down such that only the Sears Tower, or even just the observation deck of the Sears tower is being simulated.

3.4.2 Location Based Data Management

Another requirement that we have placed on our environment description is that it allow for, and assist in, the scalability of RF data. When dealing with RF systems that can potentially span entire cities or states, the amount of precalculated RF information can be staggering. A common source of precalculated RF information is pathloss between a fixed RF device and all other points within range of that device. When working with systems that represent only a relatively small portion of a city, the amount of precalculated data is easily manageable. However, for systems that span entire cities, cities that potentially contain hundreds of such devices, the amount of associated data can quickly become difficult to manage.

The hierarchical approach, though, facilitates in the management of this data by providing a convenient partitioning boundary. By splitting the pathloss data at the boundaries of each object in the hierarchy, we can control which data is loaded at any given point in the simulation. If there are no users in the Sears Tower, then we simply do not load the pathloss data for the Sears Tower, thereby freeing memory that could potentially be used by some other part of the simulation. Heuristics can also be used to further optimize what RF data is currently loaded and to know when to load data that is likely to be needed soon.

4 MOTION

With a wireless system simulator, the main requirement is that users are moving. It is of no great concern if users pass through each other as it does not affect the communication statistics. The users, though, exist in a simulated physical environment and should not be allowed to "walk" through buildings etc. The buildings and other obstructions must exist in the simulated environment as they affect the received signals and thus the ability of simulated users to receive and decode data. Moreover, it is often the case that a real city must be modeled and simulated in order to determine the effectiveness of a proposed communications network. Here the simulated city and the users within must faithfully reflect the actual city.

The wireless system simulator collects data regarding the system performance when a user population is moving and using wireless equipment. The user population, like a normal population, must have different characteristics. There must be pedestrians, slow vehicles and fast vehicles, and possibly other types as required (office based users for example). There must be users in buildings as well as groups of users moving together. The motion system must be able to handle all of these situations. Fortunately, the simplifying condition that users can be considered to be points (i.e., user-user collision detection is not required) makes the task much easier. This is important for a number of reasons which will be described in more detail shortly. However, the key reason is that it removes the necessity for a centralized controller opening the door for a more effective distributed motion system.

4.1 High Level Motion Modeling Options

As with any motion system, there are a number of ways to move users around. We could choose to divide the areas on which the users move into tiles and move from tile to tile at discrete points in time. These points in time could be regularly spaced, where different speeds would require moving over more or less tiles. Or, speed differences could be determined by moving over the same fixed distance but at different points in time.

However, as has already been discussed, the environment is described in terms of polygon areas and vector streets. This is for two reasons: 1) it makes for a more intuitive description of the environment, and 2) based upon previous simulator requirements, it has been shown to be useful as it allows a mix of RF data sources within the simulator, e.g., ray traced and/or statistical (for large areas), and measured (for small areas where a user should have movement restricted along a vector). It is desirable that the same method for motion on a vector and a polygon be employed. As motion along a vector path is relatively simple, it is desirable that movement on an area make use of a vector type motion. The main difference between vector and area motion is that the user has the opportunity to make direction changes in addition to speed changes after each movement. How often the choices can be made is dealt with in the next section.

4.2 Motion on Vectors

A simplified vector environment is shown in Figure 6, with an enlarged view following in Figure 7. In the vector environment shown above, each of the vectors knows to which vector(s) and to which end of the vector each of its ends is connected. In the above environment, there are no direction restrictions on any of the vectors, however, mobility requirements, as previously described in Section 3.1.1.1, are present. So, if a pedestrian user starts on vector v_1, its possible path is $\{v_1, v_2, v_3\}$. If a vehicular user is dropped on v_4, its possible path is $\{v_4, v_5, v_2\}$. However, if a user with mobility characteristics {pedestrian, vehicular} is dropped on any of the vectors, there are many paths that may be taken. But how does the motion take place?

The user is considered to be at the start position when the time is one second. The question is how often and how

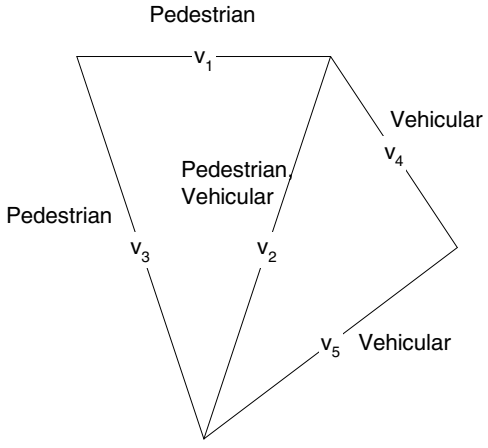

Figure 6: Simplified Vector Environment

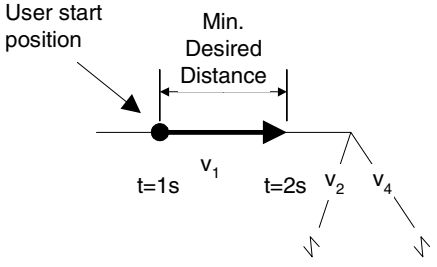

Figure 7: Unrestricted User Motion on a Vector

far should a user move in each step? As this is a wireless simulator, there is no point in moving less than a quarter of a wavelength as the change in RF signal will be insignificant. There will therefore be a minimum distance that the user should move at each step as shown in Figure 7 above. From past experience it is useful to try to move the users at least one meter in each move. So, if our user is moving at 1m/s, and the minimum desired step is 1m, then the next move would be one second later, i.e., at two seconds.

The question arises: what happens if the next move would cause the user to move beyond the end of a vector? As has been discussed before, the simulator makes use of a discrete event engine. Thus it is possible to schedule user moves at any arbitrary time in the future. The motion system would simply schedule the user to be at the end of the vector at a time appropriate for the speed. So, if in Figure 7 above, the user was 0.5m from the end of the vector at a time of two seconds, as the user is moving at 1m/s, then the next move would place the user on the end of the vector at a time of 2.5s. This is shown in Figure 8.

At the end of the vector, the user must decide on the next course of action. As described earlier, each vector has knowledge of its connections. It is therefore able to obtain details of the connected vectors from the environment system and determine matches between its own mobility characteristics and those of the attached vectors. If multiple suitable choices exist, there are a number of options open

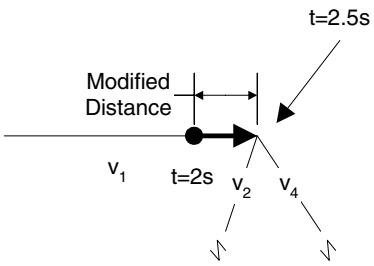

Figure 8: A Shortened User Step to the End of a Vector

to the user. It could maintain the current mobility characteristic or it could change it if it hasn't been changed for a long period of time. How it chooses is less important than the fact that it can. From a wireless simulation point of view, this ability to change mobility characteristics during the simulation allows for a diversity in the simulation population that will more realistically represent the scenario being simulated.

In the description above, users were considered to be moving at a constant speed. However, each of the mobility characteristics used includes minimum and maximum speeds and accelerations which are used to provide randomness in the motion. With each of the motion types available, different speed and acceleration values will be used to provide a model of that type of user.

4.3 Motion on Areas

Motion on areas represents a slightly different, and more difficult problem than that of motion on vectors. Whereas vectors have knowledge of their connections, areas have no knowledge of adjacent areas and users must interrogate the environment to determine adjacencies. However, the problem is simplified as the environment is arranged in a hierarchical manner. Figure 9 below shows a simple three element area environment with the corresponding environment tree shown in Figure 10.

Two users are shown in Figure 9 both on Area 2. User 1 is about to exit Area 2 via edge $e_{2,2}$. In the same way as the user moves to the end of a vector with vector motion, the user moves onto the edge which it is about to exit with area motion. Now $e_{2,2}$ and $e_{3,2}$ are the same edge and have the same vector description. As Area 2 is contained within Area 1, the user queries Area 1 to determine if there are any other areas with edges that also contain the exit point. As the exit point will also lie on $e_{3,2}$, Area 3 is a candidate for the user to move on. Mobility characteristic checks are performed to ensure that the user can cross the edge and enter Area 3. If there are no conflicts, the user will acquire Area 3 details and consider its exit point from Area 2 to be its start point on Area 3.

Considering User 2, having reached edge $e_{2,1}$, the user will perform the same checks as User 1. However, interro-

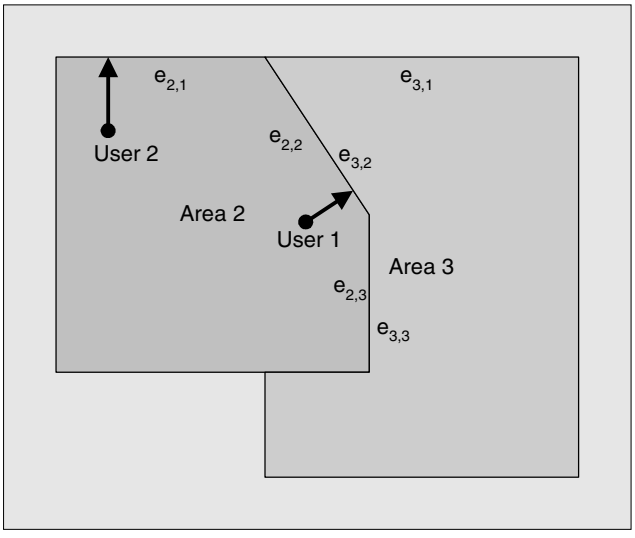

Figure 9: Simplified Area Environment

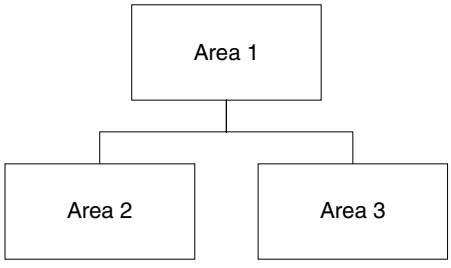

Figure 10: Environment Hierarchy Tree

gating the environment will show that no other areas have edges sharing the same point, but the exit point is still within Area 2's parent (Area 1). The user will then acquire Area 1 and, after a mobility characteristic check, will continue to move into Area 1. Should the user attempt to move out of Area 1, the environment system will not be able to find another area which shares the same exit point and the user will be reflected at the edge.

4.4 Area-to-Vector and Vector-to-Area Transitions

So far we have considered the motion on areas and vectors separately. However, the simulator must allow environments with both environment components where users may transition between both types. Motion from a vector to an area (as shown in Figure 11) can be considered relatively straight forward.

Here, the user would move to the end of the vector (as described previously). As vectors have knowledge of their end connections, the user is able to determine that the vector is connected to an area vertex and perform the appropriate check. Upon finding a suitable match of characteristics, the user may acquire the area and begin moving upon it.

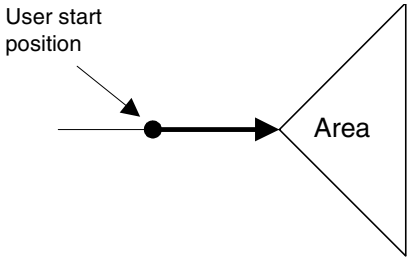

Figure 11: Vector to Area Transition

However, the situation of area to vector transition is very different. Figure 12 (below) shows a number of users moving on the same area.

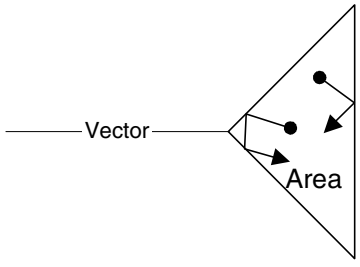

Figure 12: Area to Vector Transition Problems

Clearly, the probability of a user moving exactly onto the vertex to which the vector attached is very small indeed.

To improve the flow of users from areas to vectors, the concept of a capture edge is introduced. These are created by a pair of vertices which form an edge of the area. Once a user has touched a capture edge, it is forced to move along it in one direction only as shown in Figure 13, the idea being that the user is moved toward the exit vertex. Essentially, the capture edge is a one-way vector.

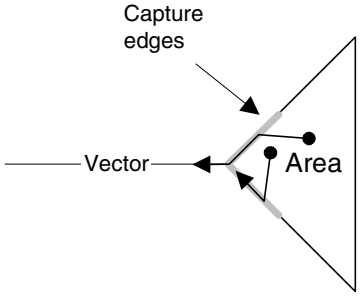

Figure 13: Area to Vector Transitions with Capture Edges

4.5 Distribution of Motion Calculations

Thus far, the specifics of the of the motion have been described without reference to actual implementation issues. As described in Section 2 and in more detail in Delve and Smith (2001), this simulation kernel is object oriented with

users essentially distributed across processors. Each user is a separate object which is free to make motion decisions independently and move based upon the motion characteristics assigned to it. Therefore, there is no single motion or physics engine within the simulator; essentially, the motion calculations are distributed across the simulator's computing resource. As described above, each of the users makes a decision on the time of next move based upon speed and required move distance. So, as the users are all making random direction and speed changes, essentially, the motion calculation is also distributed in time.

For the current requirements of the wireless simulator where users don't interact, this distribution of motion calculations is useful as it avoids the processors having to compute the motion calculations for many thousands of users at the same time. However, should the situation arise that user-user collision avoidance is required or a dynamic environment model is needed, a centralized controller will have to be included in the architecture to coordinate the calculations.

5 CONCLUSIONS AND FUTURE WORK

Motion environments designed for use with wireless radio simulators cannot utilize traditional approaches to process distribution and workload sharing. Instead, the designers must find a way for the motion environment to fit well within the wireless simulation. However, design of a motion environment cannot be an afterthought. We have found a suitable approach to motion simulation within wireless environments that successfully scales inside of these restrictions.

We have presented a method of describing physical environments suitable for use with a discrete event wireless communications system simulation. A system of motion has also been described that utilizes the physical description. It has been shown to be flexible in the types of environments that can be described. It also provides a mechanism for scaling the size of the environments to include both more and less complex descriptions. The wireless system in particular has shown its flexibility through its use of discrete event technology.

Using this fundamental motion environment, we hope to extend it to simulate new aspects of our world such as mass transit. We also hope to use this system to research the affect of mass user movement, such as flocks of users as they move towards and away from public sporting events, on wireless communications systems.

REFERENCE

Delve, T. J., and N. J. Smith. 2001. Use of DaSSF in a scalable Multiprocessor Wireless Simulation Architecture, In Proceedings of the 2001 Winter Simulation Conference, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 1321-1329. Piscataway New Jersey: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

NATHAN J. SMITH is a Software Research Engineer with Motorola Labs. He received his B.S. from Illinois State University in 2000. He has worked as a lead applications developer for Illinois State University and an embedded systems developer for Motorola. His email address is <Nathan.Smith@motorola.com>.

TREFOR J. DELVE is a Communications Research Engineer with Motorola Labs. He received his B.Eng (Honors) degree from the University of Birmingham, U.K., in 1991. He has worked as a communications engineer for The MathWorks, a systems engineer for NEC and a research associate working on underwater communications for the Ministry of Defence, U.K. His research interests include channel coding and propagation modeling. His email address is <Trefor.Delve@motorola.com>.