31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

Using Elements of Game Engine Architecture to Simulate Sensor Networks for ElderCare

Chad Godsey and Marjorie Skubic, *Member, IEEE*

Abstract— **When dealing with a real time sensor network, building test data with a known ground truth is a tedious and cumbersome task. In order to quickly build test data for such a network, a simulation solution is a viable option. Simulation environments have a close relationship with computer game environments, and therefore there is much to be learned from game engine design. In this paper, we present our vision for a simulated in-home sensor network and describe ongoing work on using elements of game engines for building the simulator. Validation results are included to show agreement on motion sensor simulation with the physical environment.**

I. Introduction

WE are using sensor networks to evaluate the daily and long-term behavior of people living in eldercare. This is done via a unique collaboration between a privately owned independent living complex and the University of Missouri [1]. This relationship allows research to utilize the facilities of this complex to develop systems for "aging in place", the idea that seniors can live independently with the aid of non-intrusive monitoring technology. Several apartments in this complex have been set up with wireless sensor networks either for testing or collecting data on voluntary participants. Using these systems, we are looking for methods of early illness detection and adverse event recognition. These systems range from monitoring activity level through motion sensors [2] to detecting falls using silhouetted video capture [3].

Up to this point, in order to generate test data we have installed one or more networks in residential apartments then either had real residents live with a sensor network for a period of time, or acted out specialized tests ourselves to acquire data with a known ground truth. This is a slow and sometimes cumbersome task and is not suitable for recreating sensor data with a specified condition over long periods of time (e.g., several months).

To address this problem, we are developing a sensor network simulator. Several simulator solutions exist in this realm, including wireless topology simulators [4], full 3D animation environments [5], and simulators for robots [8]. However, none of the existing simulation systems address our vision of a simulator for an in-home sensor network.

Manuscript received April 7, 2009. This work was supported in part by the National Science Foundation Grant IIS-0428420.

C. Godsey is with the Computer Science Department, University of Missouri, Columbia, MO 65211. M. Skubic is with the Electrical and Computer Engineering Department, University of Missouri, Columbia, MO 65211.

Our vision includes a simulation environment capable of performing the same tasks as our most common sensors. The output of this simulated network must match the output of our real network. It must be easy to switch out room layouts and change sensors for positioning and adaptations. The simulator should include support for multiple input types, such as direct input via keyboard, translation from real networks, and output from a behavior generation system which supports the generation of long term sensor data for specified behavioral conditions and/or changes.

Our needs require a system providing a complete workflow for establishing behavior scenarios through simulating behavior in a passive sensor network on to a custom logging system. Most solutions available only cover a portion of this workflow and may or may not lend themselves to integration within a larger system. Some studies have been done with game environments in evaluating flexible architectures for use in general simulation [6]. The wide variety of techniques used in game engine architectures allows for high customization of a real-time environment. It is with this notion that we are pursuing the development of a simulator with elements of game design in mind.

II. The Physical Network

The current version of our sensor network is detailed in Figure 1 [9]. It is composed of a data logger that keeps track of the passive sensors, an event-driven video network (in development, not currently installed), and a reasoning engine to fuse sensor and video data to analyze patterns of activity.

There are currently 20 apartments with this network installed at TigerPlace (without the video sensor network). They are equipped with motion sensors, a bed sensor, and a stove temperature sensor. The motion sensors are used for room activity and for specific areas such as the shower, kitchen drawers, refrigerator, and laundry closet. The sensors meant for specific areas are placed in ways to limit their view, such as being placed on the ceiling looking directly down on the shower. The bed sensor is a pneumatic strip that captures presence in bed along with qualitative pulse, respiration and bed restlessness information [7].

The portions of this network that we are concerned with initially in the simulator are the passive motion sensors and the data logger. The logger collects output from the sensors and records these events with a date and time into a database.

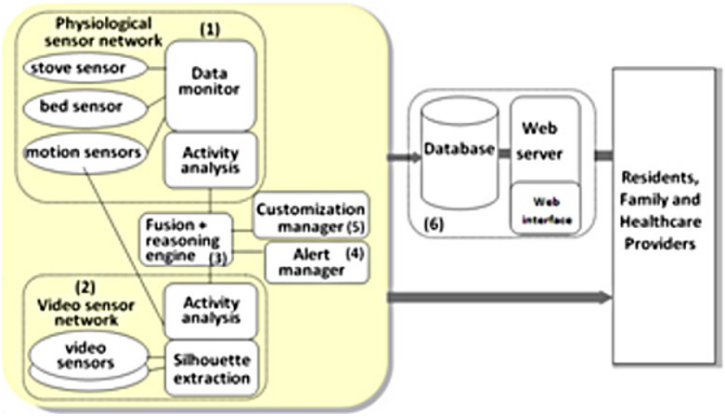

Fig 1. Organization of the sensor network used in TigerPlace. Here, we describe the simulation of the motion sensors.

III. Simulator Design

A. Intended Uses

We envision a sensor network simulator that facilitates research through multiple functions. The following discussion lists several scenarios in which the simulator will be used.

For small specific testing, the simulator should handle real-time interactive input, e.g. driving a simulated resident through the environment using keyboard strokes. With this, researchers can collect sensor data on specific activities, such as walking through the apartment from the front door to the bedroom. To further aid this use, the simulator should record input to replay at a later time. This means that a user can perform some actions and then have those actions replay over and over to collect more data on them.

For complex behavior or long term behavior, a higher level function is required. This new functionality should be able to take in a high level linguistic summary of behavior with changes over time and then build trajectories using this behavior, which the simulator can use. These behaviors will include, but not be limited to: daily routines such as time waking up, using the bathroom, preparing meals, eating, watching TV, reading, leaving the apartment as well as lifestyle patterns showing signs of depression (e.g. more sleep and limited activity during the day) or cognitive decline (e.g. wandering or pacing).

The output of the physical sensor network consists of database entries corresponding to a particular sensor with a time of fire. However, the database entries carry little to no context. They can only tell us what room activity took place and when. To better understand this output, a graphical representation of a person walking around the apartment would provide more context. By re-mapping these entries back into a spatial representation of the apartment, we can build a trajectory list to emulate a person's movement around the apartment. Visualizing this movement in the simulator can give care providers an idea of what the resident has been doing in order to better understand their state of health. Such a visualization would also aid researchers in developing algorithms to automatically analyze unusual patterns of activity.

The room layout functionality of the simulator will also facilitate preliminary testing of sensor configurations in the apartments before actual installation. By placing the motion sensors in the simulator on a particular floor plan, we can see the coverage area. This will allow analysis of overlap and dead zones, which can lead to poor data collection. Coupled with the ability to simulate complex behaviors, initial tests may be run to ensure that a sensor configuration is adequate to capture long term behavior patterns.

B. Design Overview

Figure 2 shows the generalized connectivity of our proposed solution. Currently, we are focusing on the simulation portion; behavior generation will be included in future work. The interactive portion of the simulator is being built using C++ with two open-source game development libraries; Gosu (platform independent rendering and input polling) and box2D (2D rigid body physics) among other libraries for data serialization and storage. It incorporates several techniques and components common in game engines. For flow control, there is the game loop, which is a timed loop that allows control of simulation speed. For interactive movement, a physics system is used. The physics system handles collision detection and response for movements such as running into walls and various other spatial calculations. The simulation is also data-driven using text based configuration files. These configuration files tell the simulator what objects to simulate and how.

C. Game Loop (Control)

The interactive simulator relies on a timed loop. This loop is run 60 times per second through monitoring. It controls all the logic and rendering functions. By handling the loop in this way, the simulation can be run on any modern CPU without having speed-up or speed-down issues. This separation between real time and simulation time also allows us to simulate faster or slower than real time to either build more data (faster), or view the simulation in detail.

The control schema of the loop is listed below:

- Collect input
- Step the physics simulation with time passed

- Update all simulation objects with input and time passed
- Render all simulation objects (if needed)

This method keeps all objects synchronized to the same timeframe and ensures that during any drawing operation, all objects are up to date. The parameterization of time passed allows the simulation to run in real-time, fast time or even slow time. In this way, the simulation can be used interactively, for generating large amounts of sensor data in a short amount of time, or for deep analysis of specific actions.

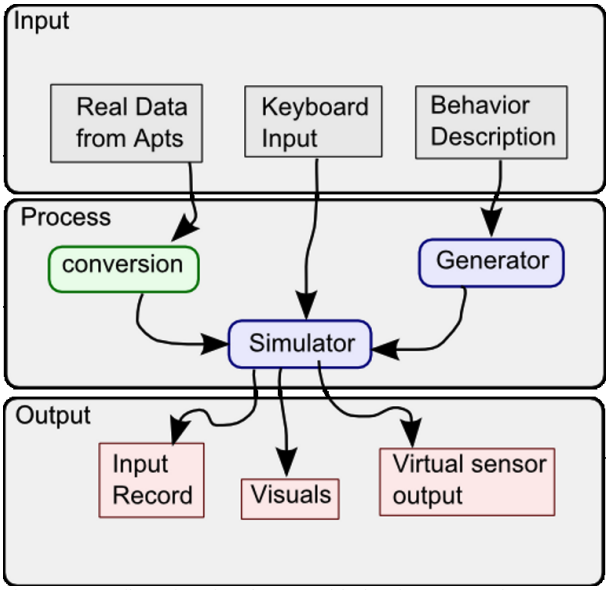

Fig 2. Data flow for simulator and behavior generation systems.

D. Physics (Interactivity)

The simulator uses rigid body physics. This means that it is intended to simulate Newtonian physics on rigid, convex hulls and assumes no intra-object movement. For the interactive portion of the simulator, this is used to handle collision detection and response for the agent's movement. In all cases it is used for ray intersection tests to determine object penetration of a motion sensor's field of view.

The primary motion sensor used in our apartments is a passive infrared (PIR) sensor that detects movement of thermal bodies. These sensors have a conical view area with a slightly higher resolution in the central vision over peripheral vision. We have found them to have view angles of 145° on the horizontal axis and 50° on the vertical axis. They fire in a timed fashion; an initial fire will occur 1-2 seconds after detecting movement and thereafter every 6-8 seconds where motion is still detected.

Simulating this device's behavior can be done in several ways. The method used here works under the assumption that all moving objects in the scene are thermal and are large enough to trigger the sensor when moving. On each update, a simulated sensor will shoot out a number of rays inside its view cone. These rays will return their length which will be shorter if they intersect some object. When this length changes, the sensor will enter the active state and fire.

Figure 3 shows the visual representation used in the simulator, where the lines indicate rays cast to test for intersection of objects in the scene. While active, the sensor will continue to fire every 6-8 seconds as long as motion is detected. This is done though a counter utilizing the game loop that chooses a random number between six and eight to wait until the next fire time.

Some PIR sensors are placed on the ceiling pointed straight down to focus on a specific area. For these, the simulator establishes a rectangular area that when intersected will enter the active state. These areas are separate from any collision response code so they do not interfere with movement.

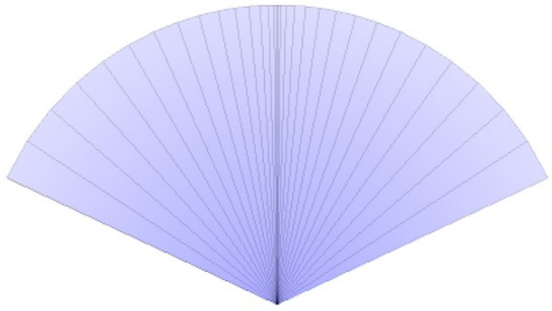

Fig 3. Graphical representation of a PIR motion sensor with test rays to detect motion. The distribution of the rays is concentrated for central view and sparse for peripheral view.

E. Text Configuration (Data)

To provide data to the simulation, a standard format should be chosen. This format must be able to represent common types of data, including text, numbers, lists, and named groups. There are several formats for text representation of data. A few examples are XML [10], JSON [11] and YAML [12]. For this simulator, JSON was chosen for its balance in readability and simplicity in data representation.

The use of text files for configuration delays data dependency until runtime instead of during compilation. This provides a standard way to pass data between systems and gives users control over how the simulation runs. The configuration scheme used for the simulator is a two level hierarchy. At the top, there is a file reference hardcoded in the simulator wherein the next level of configurations are linked. The three JSON files below this level provide detail on the environment layout, metadata for the database, and information on the person to be simulated. Each of these can be modified by separate tools, which are planned for development at a later time.

IV. Simulator Evaluation

Currently, the simulator has the core functionality for its three types of input. It will accept interactive input via a keyboard, record and play back this input, and also read a path description file then act out that path within the environment. Alongside each of these input types, the

simulator has the aforementioned implementation for PIR sensor simulation.

To test the system, we conducted several activity exercises within one of the apartments using the installed network, then translated that movement into our path description format. This activity was designed to establish patterns of firing for different speeds and distances from the sensors. One person walked perpendicular to a single PIR sensor at distances of five and ten feet in 5 minute bursts. The speeds used were about half normal walking speed, normal walking speed and double normal walking speed (roughly 0.5, 1, and 2 meters per second).

As the real data involves many variables that are not yet implemented or tuned within the simulator (e.g. sensor inaccuracies, lost or corrupted signals and other noise) we wish to establish two things about the sensor output of the simulator; first is that the simulator output resembles the output of the real network within reason, and second is that when removing all variables the simulator will produce consistent results with itself invariant of simulation speed.

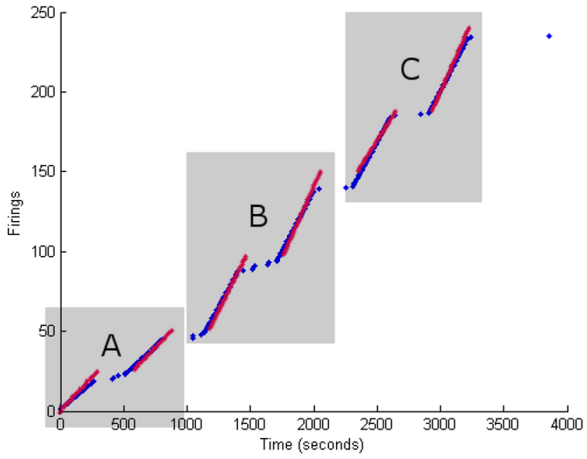

Fig. 4. Firings comparison between real data and simulation. Red is simulator output and blue is real data.

Our results for evaluating the first concern were fairly good. Once the simulated path matched the real path we found that the agreement in number of sensor firings per activity was around 93% (difference of 16 in 230 total firings). Figure 4 shows the results where portion A denotes the slow walking pace, B average, and C fast pace activities.

To address the second concern, we ran the simulator without any randomness at several speeds; five times, ten times, fifty times, and one hundred times normal speed. We found a one-to-one agreement in output between normal speed, five and ten times normal speed. For the fifty and one hundred times speed we found some disagreement dependent on the walking pace. For the slow pace the high simulation speeds lost one firing and for the fast pace two firings were gained on average. This seems to indicate that the simulator loses data when run at such speeds. Some of this data is regained when running the simulator at a faster update interval.

V. Conclusion and Future Work

A complete system for simulation and behavior generation is needed to aid our long-term research with enabling independent living for seniors. This system will take ideas and designs from several sources including robotics and game design. We are currently finishing the implementation of the simulation portion. In designing the behavior generation processes, we intend to enable end-user control through a high-level scripting language allowing a multitude of behavioral designs including reactive [13] and hybrid [14] robot architectures.

Acknowledgment

The authors would like to thank Julian Raschke for developing the minimalist game framework Gosu, Erin Catto for the rigid physics library Box2D, and the MU Eldertech research team for contribution and support.

References

[1] Rantz M, Aud M, Alexander G, Oliver D, Minner D, Skubic, M, Keller J, He Z, Popescu M, Demiris G, and Miller S, "Tiger Place: An Innovative Educational and Research Environment," AAAI in Eldercare: New Solutions to Old Problems, Washington DC, November 7-9, 2008.

[2] Wang S and Skubic M, "Density Map Visualization from Motion Sensors for Monitoring Activity Level," *Proc. of the IET Intl. Conf. on Intelligent Environments*, Seattle, Washington, 2008, pp. 64-71.

[3] Anderson D, Luke RH, Skubic M, Keller JM, Rantz M, and Aud M, "Evaluation of a Video Based Fall Recognition System for Elders Using Voxel Space," *Proc. of the Intl. Conf. of the Intl. Society for Gerontechnology*, Pisa, Tuscany, Italy, June 4-6, 2008.

[4] Park, S., Savvides, A., and Srivastava, M. B. "Simulating networks of wireless sensors," In *Proc. of the 33nd Conf. on Winter Simulation*, Washington, DC, 2001, pp. 1330-1338.

[5] Zimmerlin, T., Stanley, J., and Stone, W. "A sensor simulation and animation system," In *Proc. of the 5th Annual Conf. on Computer Graphics and Interactive Techniques*. SIGGRAPH '78. ACM, New York, NY, 1978, pp. 105-110.

[6] Brandherm, B., Ullrich, S., and Prendinger, H. "Simulation of sensor-based tracking in Second Life," In *Proc. of the 7th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems: Demo Papers,* Estoril, Portugal, May 12 - 16, 2008, pp. 1689-1690.

[7] D. Mack, M. Alwan, B. Turner, P. Suratt, R. Felder. "A Passive and Portable System for Monitoring Heart Rate and Detecting Sleep Apnea and Arousals: Preliminary Validation," *Proc. Transdisciplinary Conf. on Distributed Diagnosis and Home Healthcare (DH2),* April, 2006, Arlington, VA.

[8] R. T. Vaughan. "Stage: A Multiple Robot Simulator". Technical Report IRIS-00-394, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, 2000.

[9] M. Skubic, G. Alexander, M. Popescu, M. Rantz, and J. Keller, "A Smart Home Application to Eldercare: Current Status and Lessons Learned," *Technology and Health Care*, in press.

[10] J. Boyer, IBM (formerly PureEdge Solutions Inc.) Version 1.0, Glenn Marcy, IBM 2008. Canonical XML Version 1.1. http://www.w3.org/TR/xml-c14n11/

[11] The Internet Society, 2006, The application/json Media Type for JavaScript Object Notation (JSON), http://tools.ietf.org/html/rfc4627

[12] Oren Ben-Kiki, Clark Evans, Ingy döt Net, 2008, YAML Ain't Markup Language (YAML™) Version 1.2, http://yaml.org/spec/1.2/

[13] R. Brooks, "A robust layered control system for a mobile robot", *IEEE J. of Robotics and Automation*, vol. 2, no. 1, pp. 14- 23, 1986.

[14] R. C. Arkin, "Motor schema based mobile robot navigation," *Intl. J. of Robotics Research*, vol. 8, no. 4, pp. 92-112, 1989.