U-VR Simulator: Linking Real and Virtual Environments based on Context-Awareness

Yoosoo Oh, Changgu Kang, and Woontack Woo
GIST U-VR Lab.
{yoh, ckang, wwoo}@gist.ac.kr

Abstract

In this paper, we propose U-VR Simulator which seamlessly connects entities in real and virtual environments. Using the U-VR Simulator, smart home environments can be simulated and new virtual entities can be linked to existing real entities. The U-VR Simulator simultaneously monitors and simulates entities in real and virtual environments by exploiting a context-aware architecture. The proposed approach is fast in debugging and cost-effective in developing new entities. The approach allows application developers to rapidly develop U-VR applications and extend the applications to relevant domains. [1]

1. Introduction

Simulations of ubiquitous computing environments have investigated potential scenarios in terms of sensors, services, and environments [1]. Placements of these entities (e.g. sensors, actuators, services) in a smart home environment, as an example of ubiquitous computing environments, are intertwined. Thus, to simultaneously test these entities in a real environment typically requires a great deal of time and effort. It is expensive to test the entities in the real environment, with various sensors, actuators, and a huge number of contexts [2]. Moreover, exhaustively deploying all usable entities is difficult whenever a given framework or infrastructure is being tested. Thus, adding or extending entities in a real smart home environment is not easy because of the predefined structure and high financial cost [3].

The simulation of entities in a virtual environment can reduce the financial cost, but it is hard to maintain real operations of the system, such as real-time debugging, reactions, and system behavior [1–2]. Although simulating entities in both real and virtual environments is complicated, each case has some advantages. The simulation in a real environment can have more realistic and accurate testing, and the simulation in a virtual environment is relatively more cost-effective. Accordingly, testing in both environments which have benefits of individual environments is considerable. To this end, a ubiquitous virtual reality (U-VR) environment [4] enables computing both in real and virtual environments, such that simulations of entities prepare the development of specialized applications. The benefits of combining a virtual simulator with a real U-VR system are having advantages of both environments which include real time measurement of physical objects, realistic and systematic testing by physical objects, and additionally time saving and cost-effective evaluation by simulated objects.

As a representative example of U-VR environment, we surveyed simulators in a number of smart homes. Recently, research related to smart home simulators has dramatically advanced; related works include UbiREAL [2], eHomeSimulator [3], TATUS [5], UBIWISE [6], CASS [7], CAST [8], and C@SA [9]. However, these studies only considered newly designed entities, but did not effectively utilize given real infrastructure. These works did not support the bidirectional control between established real entities and newly-added virtual entities. Mixing both entities is important to completely link real and virtual environments. Accordingly we could analyze that we need an approach for possible bidirectional combinations of real and virtual entities.

To resolve this problem, we propose the U-VR Simulator which seamlessly connects entities in real and virtual environments by exploiting context, based on the concept of ubiquitous virtual reality. Using the U-VR Simulator, a smart home environment can be simulated and new virtual entities such as sensors, actuators, and services can be integrated to existing real entities. The U-VR Simulator simultaneously monitors and simulates entities in real and virtual environments by exploiting a context-aware architecture which makes entities distribute to both environments. Also, the proposed simulator simultaneously controls real and virtual entities in both real and virtual environments in real time.

The proposed approach is fast in debugging and cost-effective in developing new entities because the simulator effectively utilizes both existing and simulated devices. In addition, our approach has advantages such as the reduction of source code and rapid prototyping by combined real and virtual environment debugging tests. Additionally, our approach allows application developers to rapidly develop U-VR applications and extend the applications to relevant domains.

[1] This research was supported by the CTI development project of KOCCA, MCST in S.Korea.

Bruce Thomas et al. (Eds.): IWUVR 2009, pp. 52–55, 2009.
ISBN 978-0-9806098-0-6

2. U-VR Simulator

2.1. Context-aware Architecture

To operate our simulator in a heterogeneous environment, we utilized UCAM (Unified Context-aware Application Model) [10] as a context-aware architecture. UCAM consists of a UCAM Sensor, a logical sensor for linking physical sensors, and a UCAM Service which manages contexts. Thus, UCAM can manage social relationships among humans, real objects, and virtual objects; for a seamless connection of these entities we developed a modified version of UCAM, and applied it to our U-VR Simulator. We modified a communication part of UCAM to link real and virtual environments.

The U-VR Simulator links real-real, real-virtual, and virtual-virtual environments. To support this linking functionality, the modified UCAM provides seamless connection between heterogeneous environments through context exchanges which are observed among UCAM mounted real and virtual entities. In other words, the U-VR Simulator connects a real environment with a virtual environment by being aware of context flows managed by UCAM. Figure 1 shows the context flow of real and virtual environments in the U-VR Simulator. In the figure, the solid arrows indicate the context flow, and all entities (e.g., real/virtual services, real/virtual sensors) can dynamically communicate with one another through this context exchange. The dashed arrows indicate service status between real and virtual services, and sensing information between real and virtual sensors.

Furthermore, by enabling real-time communication in both environments, we can verify the operation of various entities and spontaneously simulate real environments and other environments to be implemented. Unlike previous works, our simulator simulates new entities by exploiting established hardware and software frameworks; hence, our simulator harmonizes the implementation of new and existing entities. Since our simulator utilizes the modified UCAM for developers, our simulator reduces the development time required for new entities.

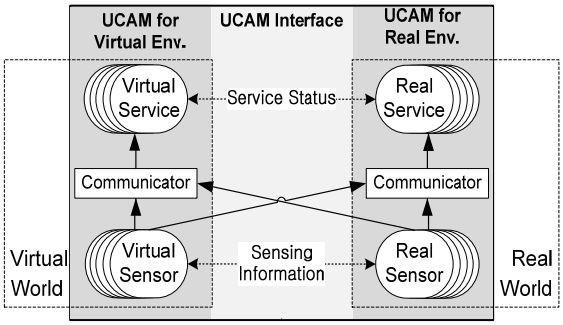


Figure 1. Context flow of real and virtual environments in the U-VR Simulator.

2.2. 3D Smart Home Simulator

The U-VR Simulator has a 3D graphical interface for a smart home environment to dynamically operate both real and virtual smart home environments by a developer. All entities in the U-VR Simulator cooperate based on UCAM, and realistically generate various contexts. Also, the entities in a virtual smart home of the simulator are connected to counterpart entities in the real smart home environment and the U-VR Simulator allows developers to add and deploy new entities.

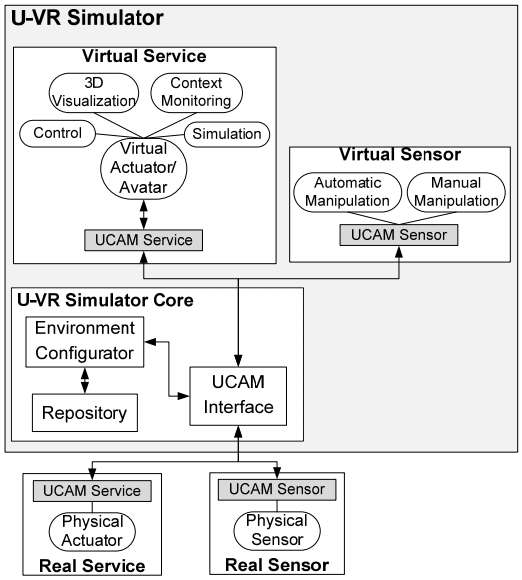


Figure 2. The implemented U-VR Simulator architecture.

The U-VR Simulator also has a UCAM Interface to ensure context awareness and an Environment Configurator to manage the 3D virtual environment. Figure 2 shows the implemented U-VR Simulator architecture. Here, the Environment Configurator configures the simulation environment which contains virtual services (including virtual actuators and avatars) and virtual sensors. The UCAM Interface interfaces with UCAM as a communication channel on a network and communicates with the Environment Configurator. The Repository records sensory data and simulator behavior, and real sensory data are automatically gathered and converted to contexts. The virtual sensor is integrated with a UCAM Sensor, and the virtual service is integrated with a UCAM Service. Also, the virtual service controls not only simulated services but virtual actuators and avatars.

The virtual sensor has two functionalities which are automatic manipulation for operating as the mirrored sensor by itself and manual manipulation for manually controlling the detection. Using the gathered data in the Repository, the virtual sensor learns characteristics and behaviors of the real sensor. For analysis, recorded simulator behavior data in the Repository is commonly used when a developer tests a smart home with no real devices; when a developer tests a real smart home, significant costs are incurred and complex

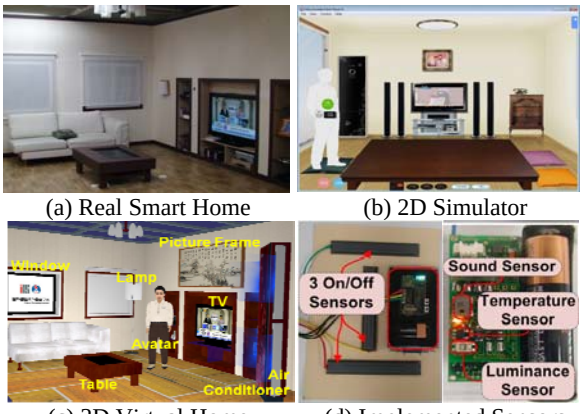
procedures are required because of the physical entities. Through the recorded system behavior, we can accurately and efficiently simulate the smart home.

The virtual service has functionalities such as control, visualization, monitoring, and simulation. The Object/Avatar Control module controls all entities in a virtual smart home, and also manages a 3D user interface for controlling each entity (avatar, object, sensor, and service). The 3D Visualization module displays the visual effects of the virtual entities; through visual animations, this module represents how entities behave based on contexts. The Context Monitoring module monitors context flow in real time such that the system behavior is also tracked to find inconsistency in processing contexts. The Simulation module constitutes the environment, and registers sensors, services, and avatar behavior. This module simulates the addition of new entities or the modification of existing entities in a real smart home. Using the simulation functionality, we can virtually simulate and test the smart home without using physical devices. Also developers can easily utilize our simulator by interfacing with UCAM.

3. Evaluation

3.1. Implementation

To verify the effectiveness of our simulator, we constructed a smart home based on the U-VR Simulator. We had previously implemented our simulator as a 2D version, built in Macromedia Flash 9 [Figure 3(b)] by modeling a real smart home [Figure 3(a)]. To enhance the visual effects and entity deployment we moved from a 2D to a 3D Simulator [Figure 3(c)]. Our 3D Simulator was developed on OpenSceneGraph [11] and 3D Studio MAX 9 [12]. Additionally, we developed our simulator on Visual Studio 2005(C++) based on cal3d [13] and the osgcal library [14].

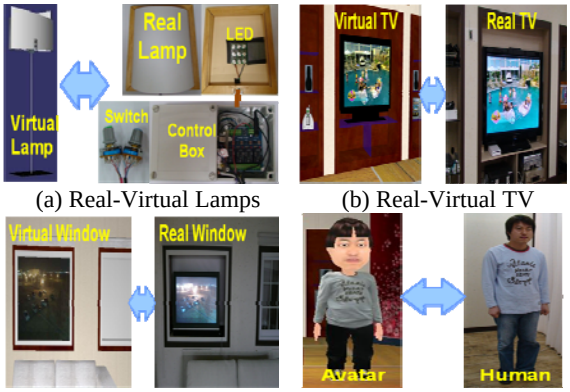


(a) Real Smart Home (b) 2D Simulator

(c) 3D Virtual Home (d) Implemented Sensors

Figure 3. Environments implemented in the U-VR Simulator.

We then implemented context-aware services with the smart sensors in a real smart home [Figure 3(a)] and its mirrored environment, a virtual smart home [Figure 3(c)], which is the reflected environment of a real smart home. We implemented real (TV, Lamp, Window, and Table) and virtual (TV, Lamp, Window, Table, Air Conditioner, Picture Frame, and Avatar) smart services. Similarly, we implemented real (profile, activity, illuminance, temperature, and physiological signal) and virtual (profile, illuminance, temperature, and location) smart sensors.



(a) Real-Virtual Lamps (b) Real-Virtual TV

(c) Real-Virtual Window (d) Real-Virtual Human

Figure 4. Implemented Services in the U-VR Simulator.

The message data format which is used to communicate between real and virtual environments is expressed by the obtained context. For instance, the message format from a virtual simulator to a real smart home environment can be represented as "*VirtualSensor:0/Name:Simulator/Illuminance:5/Temperature:21/*". Head of the format is the identity of an entity and the rest presents the used entities with each value.

The real sensors are developed as physical sensing devices based on UCAM Sensor, and the real services are also developed by physical devices based on UCAM Service. The simulated sensors are implemented according to physical aspects of the real sensor and programmed to an event-based object of a 3D simulator based on UCAM Sensor with sensor profile. The simulated services are similar to the simulated sensors. Figure 4 presents the implemented service entities. Action and response of the implemented services in a real environment are reflected to a virtual environment and vice versa. Figure 4(a) shows a real Lamp service and a virtual Lamp service as an example of our implementation; both Lamp services are interactively connected through contexts in the U-VR Simulator. Figure 4 (b) is real and virtual TV services, (c) shows real and virtual window as an information display, and (d) presents a human and a virtual avatar.

3.2. Experiment

An evaluation of the proposed simulator was conducted as follows. We experimented about the seamless connection and response of real and virtual entities in the proposed simulator. For the experimental setup, we used the following equipment: laptops for each entity (Windows Vista, Intel Core2 Duo 1.66GHz, 2GB RAM) and an illuminance sensor (Particle sensor [15]). Services integrate contexts from sensors in every 50 ms and sensors generate context in real

time. The experimented entities are a real lamp service, a virtual lamp service, a real illuminance sensor, and a virtual illuminance sensor.

Firstly, we measured sensing time and service response time between real and virtual entities. The link between real and virtual entities are four cases of virtual illuminance sensor-virtual lamp service(A), virtual illuminance sensor-real lamp service(B), real illuminance sensor-virtual lamp service(C), real illuminance sensor-real lamp service(D). The experiment measured time of 1000 responses in real-virtual lamp services covering about 1000 inputs in real-virtual illuminance sensors. For the accurate measurement without effect of network delay, we individually measured sensing time(t1) and service response time(t2), and calculated the average. Figure 5 shows the measurement result, and each case represents three average times for a virtual entity, a real entity, and their sum.

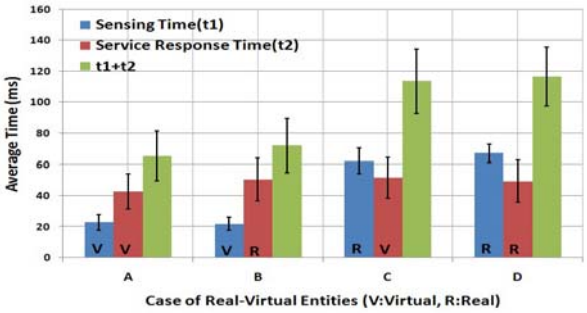


Figure 5. Average Time Measurement.

As the result in Figure 5, we could check that a total time of Case-A and Case-B was relatively small and a total time of Case-C and Case-D was relatively high. This result was the reason that sensing time(t1) of a real sensor is larger than one of a virtual sensor while service response time(t2) of all cases was not much different. In spite of different sensing time, total linking time(t1+t2) was about 65~116 ms which were very small time(cost). After all, we could know that the link process between real and virtual entities was fast. Also, we could check that our simulator operated consistently in four cases, because the sensing time of a real sensor was caused by physical sensor and then did not influence to our simulator's performance.

4. Conclusion

In this paper, we developed the U-VR Simulator that can simulate a smart home environment by enabling the addition of new virtual entities to existing real entities. The U-VR Simulator seamlessly connects entities in real and virtual environments by exploiting a modified version of UCAM. Our proposed simulator allows a developer to easily check context flow by visualization and rapidly debug each entity when problems occur. Also, our simulator is cost-effective as it uses existing real devices and by simply deploying the entities.

As a future work, we will expand our simulator to intelligently augment virtual entities into real environments. The current U-VR Simulator represents dual reality. In the near future, we will apply AR (augmented reality) technologies to our simulator and develop the simulator to overlay some entities on a real environment or a virtual environment by using a mobile phone.

5. References

[1] V. Reynolds, V. Cahill, and A. Senart, "Requirements for a ubiquitous computing simulation and emulation environment", *ACM InterSense*, 2006, vol. 138.

[2] H. Nishikawa, S. Yamamoto, M. Tamai, K. Nishigaki, T. Kitani, N. Shibata, K. Yasumoto, and M. Ito, "UbiREAL: Realistic Smartspace Simulator for Systematic Testing", *UbiComp*, LNCS4206, 2006, pp. 459–476.

[3] I. Armac and D. Retkowitz, "Simulation of Smart Environments", *IEEE ICPS*, 2007, pp. 322–331.

[4] Y. Lee, S. Oh, C. Shin, and W. Woo, "Recent Trends in Ubiquitous Virtual Reality", *IEEE ISUVR*, 2008, pp. 33–36.

[5] E. O'Neill, M. Klepal, D. Lewis, T. O'Donnell, D. O'Sullivan, D. Pesch, "A testbed for evaluating human interaction with ubiquitous computing environments", *Tridentcom*, 2005, pp. 60-69.

[6] J.J. Barton and V. Vijayarahgavan, "Ubiwise: A Ubiquitous Wireless Infrastructure Simulation Environment", *tech. report HPL2002-303*, HP Labs, 2002.

[7] J. Park, M. Moon, S. Hwang, and K. Yeon, "CASS: A Context-Aware Simulation System for Smart Home", *IEEE SERA*, 2007, pp. 461-467.

[8] I. Kim, H. Park, Y. Lee, S. Lee, H. Lee, and B. Noh, "Design and Implementation of Context-Awareness Simulation Toolkit for Context learning", *IEEE SUTC*, 2006, pp. 96-103.

[9] B. De Carolis, G. Cozzolongo, S. Pizzutilo, and V.L. Plantamura, "Agent-Based Home Simulation and Control", *ISMIS*, LNCS3488, 2005, pp. 404-412.

[10] Y. Oh and W. Woo, "How to build a Context-aware Architecture for Ubiquitous VR", *ISUVR*, CEUR-WS, 2007, pp. 032–033.

[11] OpenSceneGraph2.2, http://www.openscenegraph.org

[12] Autodesk 3D StudioMax9, http://usa.autodesk.com/adsk

[13] cal3d library, http://home.gna.org/cal3d

[14] osgcal library, http://osgcal.sourceforge.net

[15] Particle (TECO), http://particle.teco.edu/