An Architecture for Flexible Entity Configuration in a Simulation Environment*

Changgu Kang, Yoosoo Oh, and Woontack Woo

GIST U-VR Lab.
Gwangju 500-712, S. Korea
{ckang,yoh,wwoo}@gist.ac.kr

Abstract. Recently, people's interest is on the rise for virtual simulation as witnessed from a large number of applications using virtual space. Especially, simulations for ubiquitous environment focus on generating realistic data, context, contextual interpretation and have characteristics such as systematic testing, detection of rule confliction, and provision of context-aware modules. The previous works only consider systematic functionalities, but do not consider how to easily configure entities in a simulation environment for a user. To make up for these limitations, we propose a system architecture for flexible entity configuration in a simulation environment for smart space. The proposed system architecture removes dependency which is related to a parameter among functions of each entity, and independently interprets and generates context information. An application developer can implement graphical user interface(GUI) of new entities without considering the structural dependency about overall GUI of the system. Therefore, the proposed system architecture is expected to provide flexible simulation environment for an application developer to test an entity by cost effectiveness.

Keywords: Simulator, virtual reality, smart space, context-awareness.

1 Introduction

Recently, virtual space is applied to various domains such as education, entertainment, and medical service according to an increased interest of people on virtual space [1, 5, 6]. Since using the virtual space in these domains solves limited constraints such as time, technique, and cost, a user is offered with indirect yet increased educational and entertaining experiences [5]. Using virtual space, the works about simulation of smart space are also making progress. These works cover many sensors and actuators, and use acquired data to interpret context information [2, 4, 8, 12].

There are serveral related works, such as CASS [12], CAST [4], UbiREAL [8], and C@SA [2]. Most of these simulators focus on generating realistic data,

* This research is supported by Korea Culture Content Agency(KOCCA) of Ministry of Culture, Sports and Tourism(MCST) in the Culture Technology(CT) Research & Development Program 2009.

M. Chang et al. (Eds.): Edutainment 2009, LNCS 5670, pp. 38–48, 2009.

context, and contextual interpretation and have characteristics such as systematic testing, detection of rule confliction, and provision of context-aware modules. Existing simulators only consider systematic functionalities, but do not consider how to easily configure entities in a simulation environment for a user. When there is dependency between entities, an application developer has to spend repetitive effort whenever an entity is removed.

In this paper, we propose a system architecture for independent entity configuration in a simulation environment. First, the proposed architecture removes dependency among entities such as a parameter defined in programming functions . Second, the proposed architecture enables an individual entity to generate and interpret context information. Third, a developer can implement graphical user interface(GUI) of new entities without considering the structural dependency about overall GUI of the system.

The rest of this paper is organized as follows. In section 2, we explain the requirements for the proposed architecture and overall system architecture. In section 3, we describe a scenario-based implementation of simulator following the proposed system architecture. In section 4, we conclude this paper and dicuss the future works.

2 System Architecture

2.1 Requirement and Analysis

In this paper, we have considered the following requirements for the proposed system archtecture.

(1) The entities have their own context-aware module. The shared context-aware module has to confirm with changed status of all entities and generates context information according to those changes. As shown in Fig. 1 (a), the increased number of entities increases complexity and incurs synchronization problem. As shown in Fig. 1 (b), our proposed system architecture assigns context-aware module to each entity. In this approach complexity is kept to minimum and each entity is able to independently interpret and generate context information. Also there is no synchronization problem for entity configuration.

(2) There is no internal data flows between entities. The data generated by an entity are changed into context information through context-aware module. Then, all entities are able to use this context information. If this context information flows internally in a system, the flow of context is complex. Since entities are dependent in programming, a user has to remove all links related to an entity whenever he eliminates this entity. Fig. 2 (a) shows the flow of context information between related entities each other. More the number of entities exist, the complexity of a context flow is going to increase. The data of the proposed architecture flows through network. As shown in Fig. 2 (b), even though the number of entities increases, the flow of context information is kept simple and the dependency on programming is minimized because each entity has to only consider context information from the network.

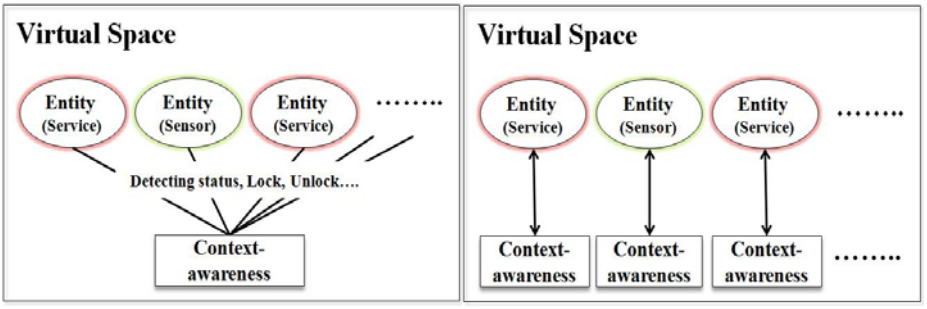


Fig. 1. (a) Shared context-aware module between entities and (b) independent context-aware module for each entity

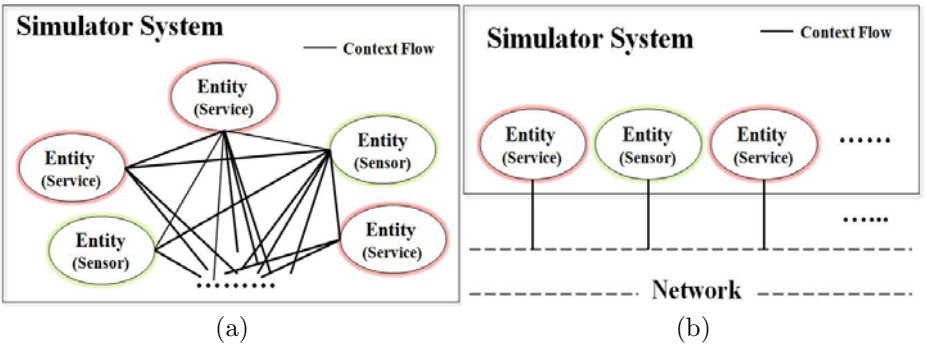


Fig. 2. The context information flows (a) internally in a system as a parameter and (b)externally through network

(3) Each entity holds own GUI. A user composes GUI for the control of entities during simulation. A user is able to implement GUI in a window or compose own GUI for each entity in accordance with the configuration. The GUI implemented in a window has a limit on the number of entities. Therefore, the latter approach is more useful for customized simulator. Fig. 3 (a) shows the window used for GUI configuration. We used this method for 2D simulator in our previous work. However our domain is smart space in ubiquitous computing environment and there are many sensors and services. Therefore, there should be no limit about the number of entities. Since the proposed system architecture provides components for GUI configuration, our architecture can guarantee independent GUI configuration and a developer can implement a new GUI without considering the structural dependency about overall GUI of the system. Fig. 3 (b) shows separate GUI for the entity. As shown in Fig. 3 (b), when a developer selects an entity, GUI for the entity is displayed.

(a) (b)

Fig. 3. GUI configuration (a) using a window and (b) individualized according to each entity

(4) There is a module for configurating indepently existing entities in a simulation space. Requirements (1), (2), and (3) are useful in keeping independency. Also it makes it easier for a developer to add and remove an entity and implement new entity. Although there are independent entities, we simulate these entities in a single simulation space. Therefore, as shown in Fig. 4, there is a module to make the simuation space and configure needed entities in the simulation space.

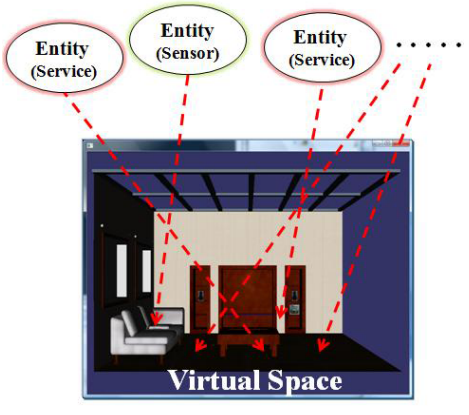


Fig. 4. Entity's configuration in virtual space

2.2 Proposed Architecture

Since the proposed system architecture keeps independency on each entity, a developer can separately implement new entity. And a developer easily add and remove existing entities whenever he organizes simulation environment. Fig. 5 shows the organized system architecture based on previously mentioned requirements. Each entity has its own context-aware module and GUI module, and there is an entity configuration module in a simulation system. Entity configuration

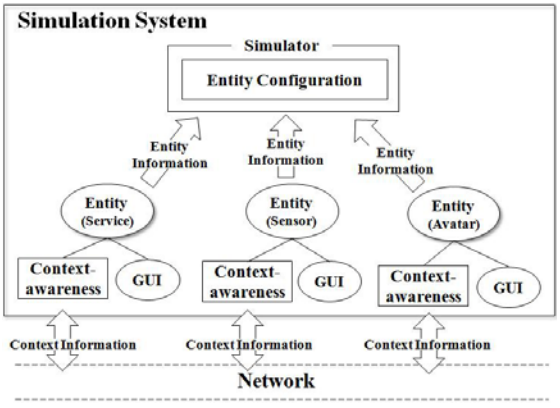


Fig. 5. System architecture

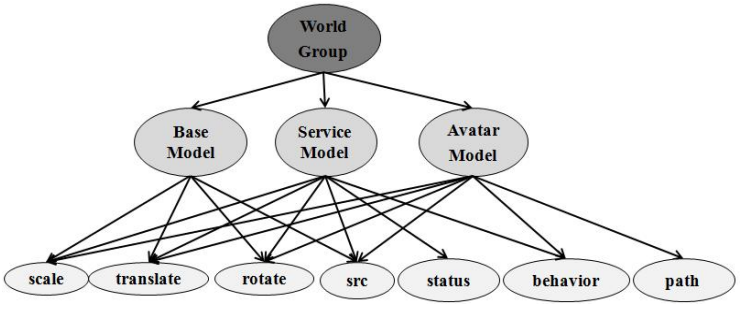
module receives entities' information to configurate a simulation environment. The communication between entities is accomplished through network.

We use Unified Context-aware Application Model(UCAM) [9] to satisfy requirement (1). As shown in table 1, we define serveral components and classes. Main components are *Communicator, Context Monitoring, GUI, and Environment Configuration*. We modified UCAM in parts to satisfy requirement (1). Using *Context Monitoring*, a user can observe context information from UCAM which is independently assigned in real-time. To satisfy requirement (2), we use existing *Communicator* in UCAM which is in charge of communication between entities for context information through User Datagram Protocol communication. *Communicator* sends the active message to all existing entities in local network whenever an entity is active. And the entity which receives the active message displays a join message to inform addition of an entity. To satisfy requirement (3), the proposed architecture provides *GUI* component. Using *GUI* component, a user defines the layout about an entity's GUI and a behavior activated according to each button. A layout is to arrange each button and decides basic appearance of a button image.

To satisfy requirement (4), we define *EnvironmentConfiguration* component as a module to organize independent entities in a simulation space according to developer's definition. EnvironmentConfiguration manages a 3D model of an existing entity in virtual space and provides an interface in XML format. As shown in Fig. 6 (a), a 3D model is managed as base model, service model, and avatar model according to each entity's characteristic. A base model does not define the status and the behavior of a 3D model as a static model in the simulation environment. A service model defines the status and the behavior of a 3D model in accordance with the change of the entity's status. An avatar model can define the status and the behavior such as a service model and have path. As *EnvironmentConfiguration* provides a interface of XML format in Fig. 6 (b), a developer can configurate file path, position, size, and rotation of all 3D models.

Table 1. Main component and class

Component	Class	Description
Communicator	Communicator	Responsible for exchanging context information between entities by UDP
Context Monitoring	Browser	Display context information from an entity
GUI	Control Button	Select layout according to button image
	ButtonExecutor	Select behavior of activated entity according to each button
Environment Configuration	PickableObject	Recognize composed entity
	Environmental Setup	Manages entity 3D models
		Provide XML interface for user defined environmental configuration



(a)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <EnvironmentConf version="1.0" xmlns="http://uvr.gist.ac.kr/" xmlns:xsi="http://www.w3.org/2001/
  - <Header copyright="Changgu Kang">
      This is environment configuration for simulation!
      <Simulator name="3D Smart Home Simulator" version="1.0" />
    </Header>
  - <Environment>
    - <BaseModel>
        <SRC>Model/BaseModel/Room.osg</SRC>
        <SCALE>"1.0 1.0 1.0"</SCALE>
        <TRANSLATE>"0.0 0.0 0.0"</TRANSLATE>
        <ROTATE>"0.0 0.0 0.0 0.0"</ROTATE>
      </BaseModel>
    - <BaseModel>
```

(b)

Fig. 6. (a) Structure for managing 3D model and (b) XML interface for entity's configuration

Additionally, a developer can set up the name of a service entity and profile information of an avatar entity.

3 Implementation and Scenario

3.1 ubiHome 3D Simulator

We implemented a simulator and service entities using previously mentioned components. The implemented simulator was based on ubiHome [3] which exists as a smart home in our laboratory. We named ubiHome 3D Simulator as the system prototype of the proposed architecture. A service was implemented as the form of dynamic link library, and it is possible to distribute to a developer who uses the same system. We used OpenSceneGraph Library [10] as a 3D graphic toolkit. And we used Cal3d [7] and osgcal [11] library for the animation of a 3D model. 3D models were made using 3D MAX [14].

Fig. 7 shows components and data flow of implemented services and simulator. There are additional components such as *EventHandler, EntityCore, and Visulaization* except main components. *EventHandler* manages animation according to the changes of service's status. And *EntityCore* has 3D model information about a service and manages all components of a service. *Visualization* shows services composed according to developer's definition.

We implemented virtual services such as TV, window, lamp, air conditioner, light, and avatar. TV, window, and lamp service also exist in real ubiHome. And air conditioner, light, and avatar service only exist in virtual ubiHome. As shown in table 2, characteristics of services are as follows. The services which exist together in real and virtual ubiHome are synchronized. For the synchronization, TV and window service use JPEG as the format of image which input from real TV and real window at 20 fps. To synchronize with a real lamp, a virtual lamp service has characteristics of three colors and three brightness levels like a real lamp. The brightness of virtual light can be controlled by using a real illuminance sensor and a virtual light service has thirteen levels. The air conditioner

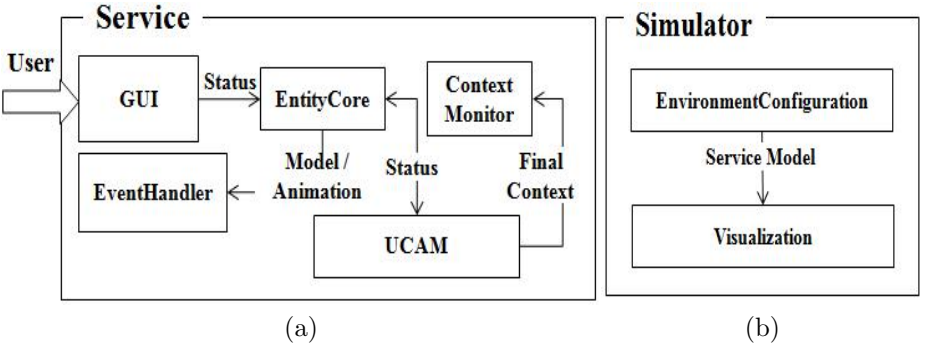


Fig. 7. The components and the data flow of (a) the implemented services, and (b) the implemented Simulator

Table 2. Service specification

Space	Service	Specification
Real ubiHome	TV, Window	JPEG Format, 20fps
	Lamp	Three color(blue, red, green), Three levels of brightness
Virtual ubiHome	TV, Window	Real time control, Synchronizing real TV and window
	Lamp	Real time control, Synchronizing real lamp, Three color(blue, red, green), Three levels of brightness
	Air conditioner	Real time control
	Light	Real time control, 13 levels of brightness
	Avatar	Real time control, Synchronizing real human using physical position and activity sensor

(a)

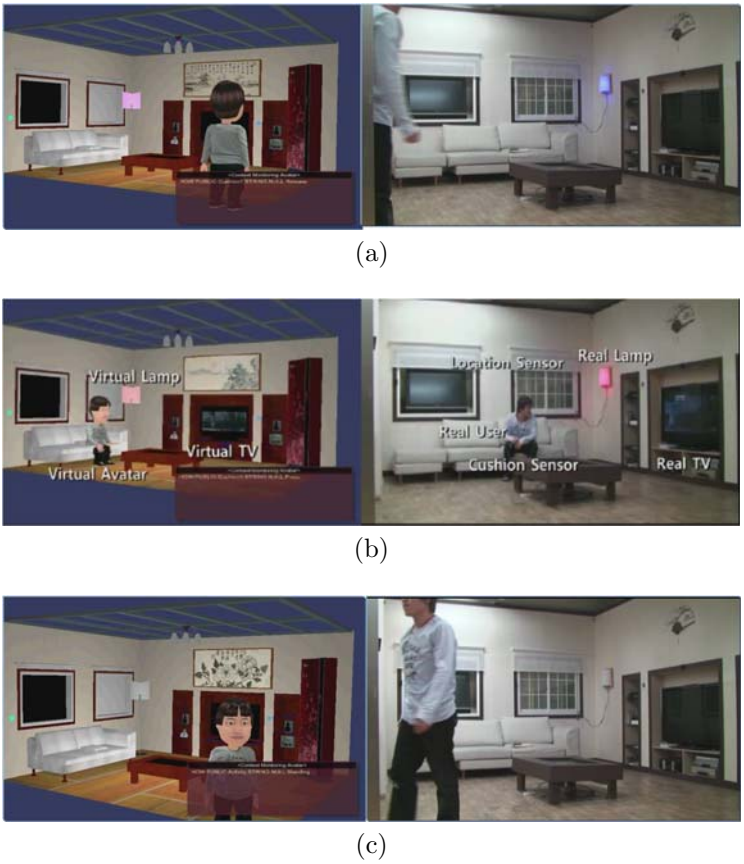
(b)

Fig. 8. (a) Real ubiHome and services, and (b) the implemented virtual ubiHome and services

service can be controlled using a real temperature sensor. An avatar has profile information such as age, name, and gender. And an avatar can be controlled by a devloper and reflects behaivor or positon of a real human using position sensors(cushion sensor, IR sensor) and an activity sensor(particlesensor [13]). The behavior of an avatar is walking, running, sitting, and standing. Fig. 8 (a) shows real ubiHome and real services, and Fig. 8 (b) shows the implemented virtual ubiHome and virtual services.

3.2 Scenario

We wrote a scenario to test ubiHome 3D Simulator. The targets of the written scenario are cushion sensor and IR sensor. The virtual services used for cushion sensor and IR sensor are TV and lamp service. Because these virtual services synchronize with real services, real and virtual services are actioned. The scenario is as follows.



(a)

(b)

(c)

Fig. 9. Demo screenshot: (a) Enter a room, (b) Virtual lamp and TV service are active, (c) Go out and all service are turned off

"On Feb 1, 2009, PM 5:00, after work, Chan enters a room. When Chan enters a room, IR sensor is active in real ubiHome. At the same time, a Chan's avatar enters a virtual room because context information of IR sensor is reflected into a virtual room. Chan moves forward to a cushion to take a break and sits down. Then the cushion sesnor detects the behavior and the position of Chan, and Chan's avatar moves forward a virtual cushion and sits down. When a cushion sensor detects the behavior of Chan, which is to sit down, a virtual lamp and a virtual TV service are activated. After taking a break, Chan goes out. When IR sensor detects Chan's leaving, a Chan's avatar goes out and all virtual services are turned off."

We defined the behavior of a service according to needed context information to be applied to our scenario. And we set up cushion sensor and IR sensor in real ubiHome. We confirmed working of rhe cushion sensor and IR sensor, and Fig. 9 is a screenshot of demonstration about our scenario. Fig. 9 (a) shows when chan enters a home. And Fig. 9 (b) shows that services are active. Final Fig. 9 (c) shows that all services are turned off.

4 Conclusion and Future Work

In this paper, we proposed a system architecture which keeps independency among all entities composed in a simulation environment and is useful for implementaion, addition, and remove of an entity whenever a developer organizes the simulation environment. The proposed system architecture removes dependency which is related to a parameter among functions of each entity, and individually interprets and generates context information. A developer can implement graphical user interface(GUI) of new entities without considering the structural dependency about overall GUI of a system.

We defined requirements for the proposed architecture and implemented ubiHome 3D Simulator and virtual services. After writing a scenario, we tested ubiHome 3D Simulator based on our architecture. Using real sensors(IR sensor, cushion sensor) and virtual services(TV, lamp), we confirmed our system. As the future work, we are considering to apply our architecture to other domains and confirm the usefulness of our architecture.

References

1. Alverson, D., Saiki, S., Caudell, T., Panaiotis, K., Sherstyuk, A., Nickles, D., Holten, J., Goldsmith, T., Stevens, S., Mennin, K., et al.: Distributed immersive virtual reality simulation development for medical education. Journal of International Association of Medical Science Educators 15, 19–30 (2005)
2. De Carolis, B., Cozzolongo, G., Pizzutilo, S., Plantamura, V.: Agent-Based Home Simulation and Control. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS, vol. 3488, pp. 404–412. Springer, Heidelberg (2005)
3. Jang, S., Shin, C., Oh, Y., Woo, W.: Introduction of'ubiHome'Testbed. IPSJ SIG Technical Reports 2005(60), 215–218 (2005)

4. Kim, I., Park, H., Noh, B., Lee, Y., Lee, S., Lee, H.: Design and Implementation of Context-Awareness Simulation Toolkit for Context learning. In: IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006, vol. 2 (2006)
5. Kock, N.: E-Collaboration and E-Commerce In Virtual Worlds: The Potential of Second Life and World of Warcraft. International Journal of e-Collaboration 4(3), 1–13 (2008)
6. Lee, E., Wong, K.: A Review of Using Virtual Reality for Learning. In: Pan, Z., Cheok, D.A.D., Müller, W., El Rhalibi, A. (eds.) Transactions on Edutainment I. LNCS, vol. 5080, pp. 231–241. Springer, Heidelberg (2008)
7. C. library, `http://home.gna.org/cal3d`
8. Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., Ito, M.: Ubireal: realistic smartspace simulator for systematic testing. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 459–476. Springer, Heidelberg (2006)
9. Oh, Y., Woo, W.: How to build a Context-aware Architecture for Ubiquitous VR. In: International Symposium on Ubiquitous VR, p. 1 (2007)
10. OpenSceneGraph, `http://www.openscenegraph.org/`
11. osgcal library, `http://osgcal.sourceforge.net`
12. Park, J., Moon, M., Hwang, S., Yeom, K.: CASS: A Context-Aware Simulation System for Smart Home. In: 5th ACIS International Conference on Software Engineering Research, Management & Applications, SERA 2007, pp. 461–467 (2007)
13. Particle(TECO), `http://particle.teco.edu/`
14. A. D. StudioMax9, `http://usa.autodesk.com/adsk`