A Sensor-based Interaction for Ubiquitous Virtual Reality Systems *

Dongpyo Hong[1] Julian Looser[2] Hartmut Seichter[2] Mark Billinghurst[2]
Woontack Woo[1]

GIST U-VR Lab.[1]
Gwangju 500-712, Korea[1]
{dhong, wwoo}@gist.ac.kr[1]

HIT Lab NZ, University of Canterbury[2]
New Zealand[2]
{julian.looser, hartmut.seichter, mark.billinghurst}@canterbury.ac.nz[2]

Abstract

In this paper, we propose a sensor-based interaction for ubiquitous virtual reality (U-VR) systems that users are able to interact implicitly or explicitly with through a sensor. Due to the advances in sensor technology, we can utilize sensory data as a means of user interactions. To show the feasibility of the proposed method, we use ComposAR as a test-bed that is a script-based interaction authoring tool for augmented reality (AR) systems. By adding sensor-based interaction features to ComposAR, a user can interact with virtual 3D contents through a sensor. We believe that the proposed method provides natural user interactions for U-VR systems.

1. Introduction

Since Weiser introduced the concept of ubiquitous computing, computing paradigm has been changed significantly from system-oriented to user-oriented. To develop user-oriented systems, there have been many research efforts on understanding the situations of users [1, 2]. In particular, the advances in sensor technology enables system developers to utilize various kinds of sensors in user interactions [3, 4]. From the perspective of contents provision, most previous works on context-aware computing applications could not overcome 2D contents. However, Augmented Reality (AR) technology convinced researchers of a complementary technology for the 2D contents provisions. As a result, many tools and frameworks for the development of AR system have been emerged such as ARToolkit, osgART, and ARToolkitPlus [5, 6, 7]. But, those toolkits demanded developers to have high quality of programming skills in order to build AR systems. To help non-technical users to build AR systems with simple configurations, many AR authoring tools have been also proposed [8, 9, 10]. Most AR authoring tools have focused on how to manipulate 3D models such as selection, position, and rotation of them with fiducial markers [11, 12, 13]. However, they are not enough to support dynamic user interactions in real environment. Therefore, there has been an activity to realize Ubiquitous Virtual Reality (U-VR) systems where users are able to interact with virtual 3D contents on real environment as well as share them with others through their explicit or implicit intensions [14].

In this paper, we propose a sensor-based user interaction for U-VR systems by utilizing scripting-based AR authoring tool, ComposAR [1], that allows the user to write interaction scripts and provides immediate runtime feedback from the interaction scripts. Due to its scripting feature, we can easily setup U-VR systems and test them on the fly. However, current version of ComposAR only allows users to interact with virtual 3D contents through the fiducial markers (See Figure 1). Thus, we need to modify ComposAR to allow sensory data as a means of user interaction. In the proposed method, the user can manipulate the loaded virtual 3D contents by moving a sensor (acceleration of $x-$, $y-$, and $z-$axis), making noise, changing the intensity of light, and even the variation of temperature. In addition, all the authored contents and states are stored in a XML file format, and restored as the same contents and states. Thus, the user can resume their interactions anytime and anywhere. Moreover, this feature makes it possible to deliver and share user interactions as well as contents with other U-VR systems.

This paper is organized as follows. In section 2, we briefly introduce the main features of ComposAR and its development environment. In section 3, we show the configuration of a sensor-based interaction and explain its implementation in detail. Finally, we discuss future works in section 4.

*This research is supported in part by the Foundation of UCN Projects, the MKE, and the 21C Frontier R&D Program in Korea as a result of subproject UCN 08B3-O1-20S and by the FRST New Zealand founded research project CALMARS at the HITLab New Zealand

[1] http://www.hitlabnz.org/

2. ComposAR: AR Authoring Tool

The main features of ComposAR are: association of fiducials and 3D content, dynamic modification of the properties of 3D contents, live interaction scripting, persistant storage of content and configuration. Figure 1 illustrates a typical development environment of ComposAR.

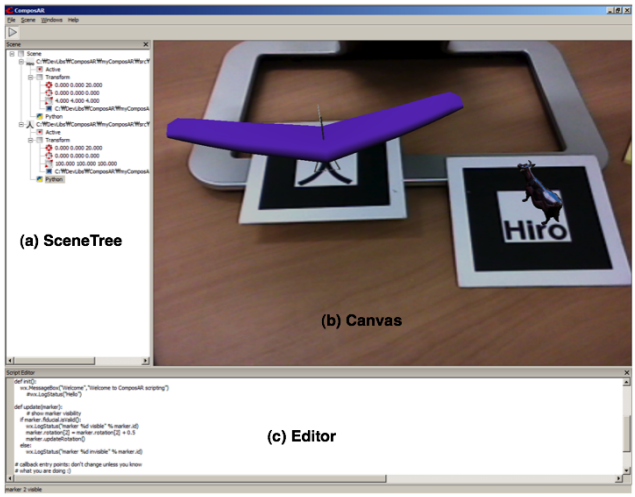


Figure 1. The environment of ComposAR

As shown in Figure 1, the development environment of ComposAR has 3 panes and each pane can be floatable except the actual rendering canvas. (a) *SceneTree* indicates how the current scene graph is organized with contents like markers, models and interaction scripts. (b) *Canvas* is an OpenGL rendering context showing live video and to render 3D contents. (c) *Editor* shows interaction scripts. Similar to conventional authoring tools, ComposAR also provides a menu and a toolbar.

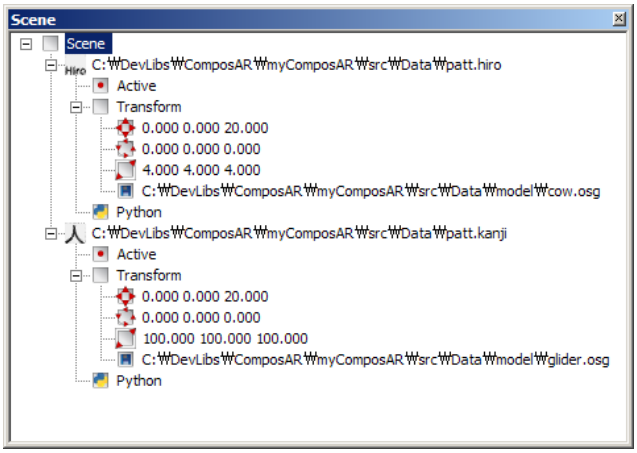


Figure 2. The SceneTree of ComposAR

In dynamic association of markers and 3D contents, users can click *SceneTree* or choose menu which marker they want to associate with which 3D contents. Figure 2 shows an example of currently associated markers and 3D contents (See Figure 1). Through the *ScenTree*, users are also able to dynamically change the properties of the loaded 3D contents. For instances, users can locate 3D contents a certain position from the marker, rotate and scale the 3D contents.

3. Sensor-based Interaction Script Implementation

In this section, we show a sensor-based interaction configuration and reveal the implementation in details. Figure 3 shows sensor-based interaction configuration.

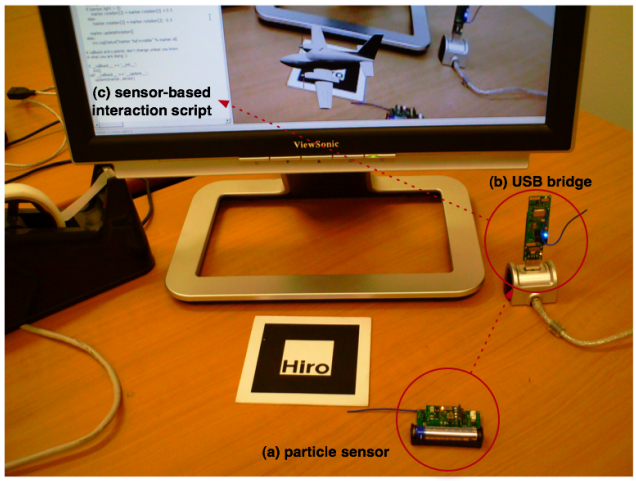


Figure 3. Sensor-based Interaction Configuration

As shown in Figure 3, we utilize a particle sensor [2] which can acquire the values of acceleration of $x-$, $y-$, and $z-$axis, sound, light and temperature. In order to extract the acquired signals from the particle sensor, we need its receiver where we use a USB interface receiver. From the receiver, we can selectively extract any of data from the particle sensor. The extracted data finally are delivered to a sensor module in ComposAR, which are mainly used in the interaction scripts. Figure 4 shows the sensor-based interaction configuration within ComposAR environment.

Before we explain sensor-based interaction script, we review the fiducial marker-based interaction script in ComposAR. Typically users can change certain behavior of the loaded virtual 3D contents by checking whether a marker is visible or invisible as follows.

[2] http://particle.teco.edu

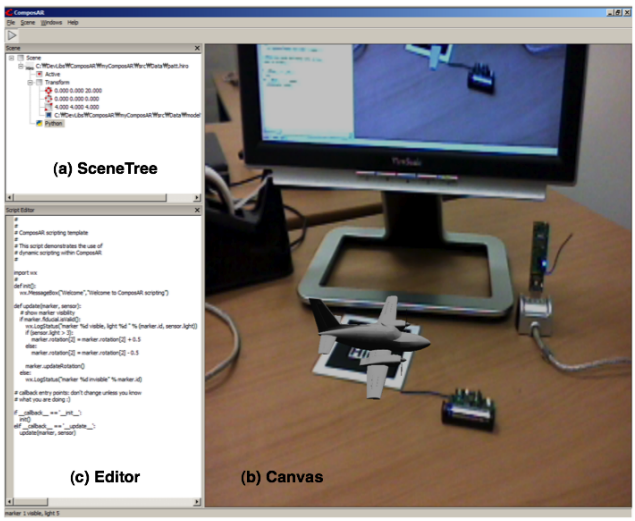


Figure 4. Sensor-based Interaction

```
import wx

def init():
    wx.MessageBox("Welcome", "Welcome to ComposAR scripting")

def update(marker):
    if marker.fiducial.isValid():
        wx.LogStatus("marker %d visible" % marker.id)
        marker.rotation[2] = marker.rotation[2] + 0.5
        marker.updateRotation()
    else:
        wx.LogStatus("marker %d invisible" % marker.id)

if __callback__ == '__init__':
    init()
if __callback__ == '__update__':
    update(marker)
```

As shown in the above code, the loaded 3D contents is rotating 0.5 degree along z-axis when the marker is visible. In particular, ComposAR provides two **callback** functions such as `__init__` and `__update__`. In `__init__`, users can initialize their interactions. In `__update__`, users can make more interesting interactions with 3D contents. Because current ComposAR only allows the users to write interaction scripts according to the properties of a marker, we need to modify *update(marker)* function. In addition to the modification of *update()* function, we also need to add a sensor module to acquire sensory data in ComposAR. As a result, we added Python-enabled *ParticleSensor* module on top of Python-binding particle library which can utilize a particle sensor as follows.

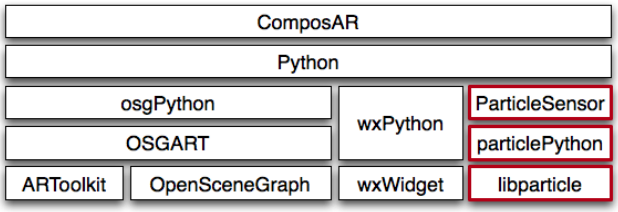


Figure 5. Sensor-based Interaction Module

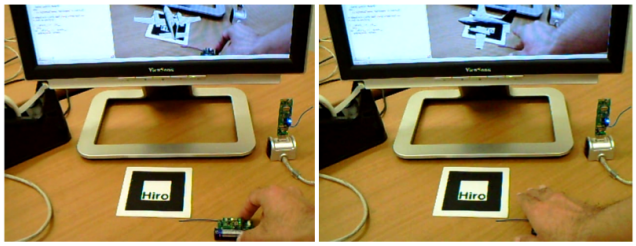
As shown in Figure 5, ComposAR mainly utilizes wxPython [3] for wxWidget (it supports convenient GUI libraries) and osgPython [4] for OpenSceneGraph [5] (it support high quality of graphic rendering and scene graph management). Meanwhile, OSGART [6] supports ComposAR to load video, tracker, and scene rendering.

With the modified ComposAR, we are able to support sensor-based script interactions in addition to marker-based script interactions. That is, the user can write interaction scripts based on the variation of sensory data as the following code.

```
def update(marker, sensor):
    if marker.fiducial.isValid():
        if (sensor.light > 3):
            marker.rotation[2] = marker.rotation[2] + 0.5
        else:
            marker.rotation[2] = marker.rotation[2] - 0.5
        marker.updateRotation()

if __callback__ == '__update__':
    update(marker, sensor)
```

For the sensor-based interaction script, we also modified `__update__` function in ComposAR to have a sensor as a parameter as well as a marker. Thus, **update (marker, sensor)** function now allows to have properties of a marker as well as values of a sensor. Therefore, users can interact with the loaded 3D contents by moving the sensor, changing lighting condition, or making noise. Figure 6 shows an example of sensor-based interaction with ComposAR.



(a) light level = 5 (b) light level = 1

Figure 6. Interaction with the light level

As shown in Figure 6, a user can control the rotation of the loaded 3D contents by controlling the level of light. In this method, we can write many interesting interactions with the loaded 3D contents.

In conventional authoring environments, it is important to store and restore currently authored data or scene. Thus, ComposAR also provides such features to users. When users are done with their own authoring, they can save the current scene data as well as they can save interaction scripts in a XML file format. When they need the saved authoring data, they can retrieve them from the XML file.

[3] http://www.wxpython.org
[4] http://code.google.com/p/osgswig
[5] http://www.openscenegraph.org

However, current version of ComposAR does not support to embed interaction scripts into the XML file format. Thus, we modified ComposAR to include interaction scripts in the XML file. The following XML code shows how the modified version of ComposAR stores current scene and interaction scripts.

```
<?xml version="1.0" ?>
<composar os="nt" utc="Fri, 11 Apr 2008 13:14:16" version="0.1">
  <scene>
    <videos/>
    <trackers/>
    <markers>
      <marker model="cow.osg" name="patt.hiro"
                  position="0.0 0.0 20.0"
                  rotation="0.0 0.0 21.0"
                  scale="4.0 4.0 4.0"
                  script="hello.py"/>
      <marker model="glider.osg" name="patt.kanji"
                  position="0.0 0.0 20.0"
                  rotation="0.0 0.0 13.0"
                  scale="100.0 100.0 100.0"
                  script="hello.py"/>
    </markers>
  </scene>
</composar>
```

In the XML file, we simply ignored video and tracker elements because we assumed that users use the provided video and tracker libraries. This feature will be revised in future release.

4. Discussion and Future work

In this paper, we proposed a sensor-based interaction for U-VR systems with the scripting-babsed authoring tool for AR systems. Unlike the existing approaches, ComposAR helps users to easily build and test a simple AR system due to scripting environment. In ComposAR, users are able to load virtual 3D contents and manipulate them dynamically. While we have kept such features, we modified and added some features to the current ComposAR. For example, users can write their own interaction scripts not only with visibility of markers but also values of sensory data. In addition, the users can save their interaction scripts with other parameters of 3D contents and markers for AR systems, and reload the same states as they saved. With the proposed features, the users are able to selectively share their contents as well as interactions with others.

As future work, we can extend this technique to natural environment so that we can author interactions not only markers but also natural objects. In addition, we need to investigate on provisions of useful interaction template scripts for ordinary users, and other interaction metaphors rather than fiducial markers. We also need to evaluate the usability of sensor-based interaction comparing to fiducial marker-based interaction.

References

[1] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," *IEEE Workshop on Mobile Computing Systems and Applications*, pp. 85–90, 1994.

[2] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *In the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000)*, 2000.

[3] A. Schmidt, M. Beigl, and H.-W. Gellersen, "There is more to context than location," *Computers and Graphics*, vol. 23, no. 6, pp. 893–901, 1999.

[4] H. Lieberman and T. Selker, "Out of context: Computer systems that adapt to, and learn from, context," *IBM SYSTEMS JOURNAL*, vol. 39, no. 3 - 4, pp. 617 – 631, 2000.

[5] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *the 2nd International Workshop on Augmented Reality (IWAR 99)*, pp. 85–94, 1999.

[6] J. Looser, R. Grasset, H. Seichter, and M. Billinghurst, "OSGART - A pragmatic approach to MR," in *Proceedings of International Symposium of Mixed and Augmented Reality*, 2006.

[7] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," in *Computer Vision Winter Workshop*, 2007.

[8] R. Dörner, C. Geiger, M. Haller, and V. Paelke, "Authoring Mixed Reality – A Component and Framework-Based Approach," in *Proceedings of International Workshop on Entertainment Computing - Special Session on Mixed Reality Entertainment Computing*, 2002.

[9] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter, "DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences," in *Proceedings of the 17th annual ACM symposium on User Interface Software and Technology*, pp. 197–206, 2004.

[10] C. Knöpfle, J. Weidenhausen, L. Chauvigné, and I. Stock, "Template Based Authoring for AR based Service Scenarios," in *Proceedings of the IEEE Virtual Reality 2005 (VR'05)*, pp. 237–240, 2005.

[11] J.-D. Yim and T.-J. Name, "Developing Tangible Interaction and Augmented Reality in Director," in *Proceedings of Conference on Human Factors in Computing Systems*, pp. 1541–1541, 2004.

[12] G. A. Lee, C. Nelles, M. Billinghurst, and G. J. Kim, "Immersive Authoring of Tangible Augmented Reality Applications," in *Proceedings of Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pp. 172–181, 2004.

[13] T. Ha and W. Woo, "Graphical tangible user interface for a ar authoring tool in product design environment," in *Proceedings of International Symposium on Ubiquitous Virtual Reality 2007*, vol. 260 of *CEUR-WS*, 2007.

[14] Y. Suh, K. Kim, J. Han, and W. Woo, "Virtual reality in ubiquitous computing environment," in *International Symposium on Ubiquitous Virtual Reality*, pp. 1–2, 2007.