

extension to support sensor-based interaction and explain its implementation in detail. Finally, we discuss potential applications and future works in section 4.

2. ComposAR: AR Authoring Tool

ComposAR is an AR authoring tool that has been previously developed at the HIT Lab NZ. The main features of ComposAR are: support for AR viewing showing virtual content overlaid on the real world, the association of 3D content with fiducial markers, dynamic modification of the properties of virtual objects, contents, live python based interaction scripting, persistent storage of the AR scene configuration. Figure 1 shows the ComposAR interface.

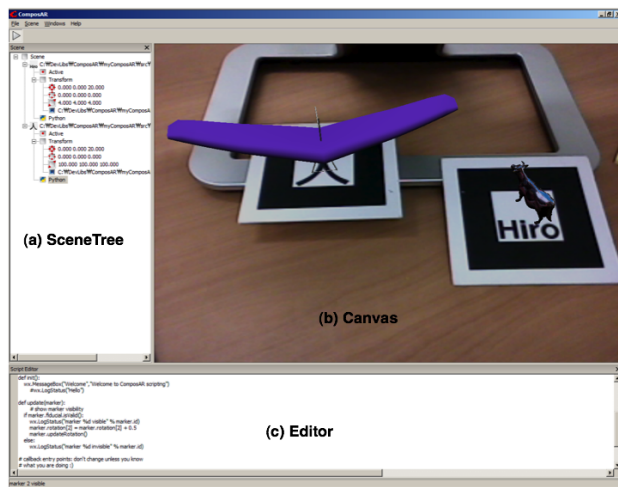


Figure 1. ComposAR Environment

The ComposAR interface has 3 panes: (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 virtual 3D content. (c) *Editor* shows interaction scripts.

To dynamically associate AR tracking markers and virtual 3D content, users can click the *SceneTree* or choose from a menu which marker they want to attach the virtual content to. Figure 2 shows an example of currently associated markers and 3D contents (See Figure 1). Through the *SceneTree*, users are also able to dynamically change the properties of the loaded virtual contents. For instances, users can translate, rotate or scale the virtual content shown on a marker.

3. Sensor-based Interaction Implementation

Although the current version of ComposAR provides an intuitive AR authoring tools it does not have support for

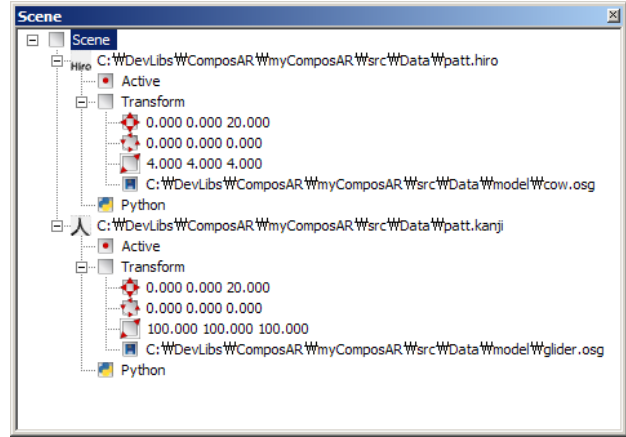


Figure 2. The SceneTree of ComposAR

external sensor input and so is limited in its ability to author U-VR applications. In this section, we describe extensions that we have made to ComposAR to support sensor input. Figure 3 shows sensor-based interaction configuration.

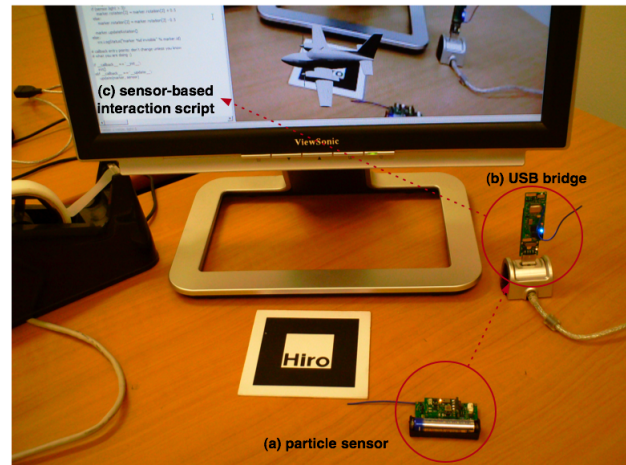


Figure 3. Sensor Interaction Configuration

In our research we use a particle sensor² which is a small hardware board that can measure acceleration (x -, y -, and z -axis), sound, light and temperature values. The particle sensor communicates these values wirelessly to a host PC using a USB interface receiver. From the receiver, we can selectively extract any of data from the particle sensor. The extracted data is then delivered to a sensor module we have written for ComposAR, which allows it to be used in the interaction scripts. Figure 4 shows the sensor-based interaction configuration within ComposAR environment.

Before we can explain our sensor-based interaction

²<http://particle.teco.edu>

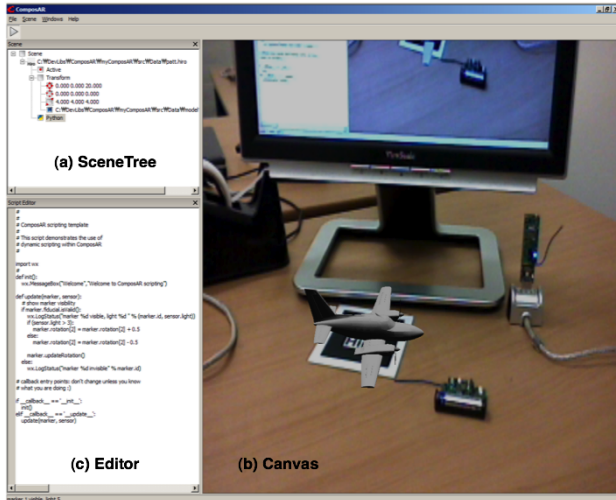


Figure 4. Sensor-based Interaction

script, we review a typical fiducial marker-based interaction in ComposAR. Using the python script below users can change the behavior of a virtual model by checking whether a marker is visible or not.

```
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)
```

In this case the virtual model is rotated 0.5 degree about the z -axis when the marker is visible. In particular, ComposAR provides two **callback** functions `__init__` and `__update__`. In `__init__`, users can initialize their interactions. In `__update__`, users can specify more interesting interactions with the virtual content. In ComposAR the interaction scripts are associated with a marker, so we needed to modify the `update(marker)` function to add support for sensor input. In addition to the modification of `update()` function, we also need to add a sensor module to read the sensory data into ComposAR. As a result, we added a Python-enabled *ParticleSensor* module on top of Python-binding particle library which can utilize a particle sensor as follows.

As shown in Figure 5, ComposAR uses wxPython³ for wxWidget support (convenient GUI libraries) and osgPython⁴ for OpenSceneGraph⁵ rendering (high quality

³<http://www.wxpython.org>

⁴<http://code.google.com/p/osgswig>

⁵<http://www.openscenegraph.org>

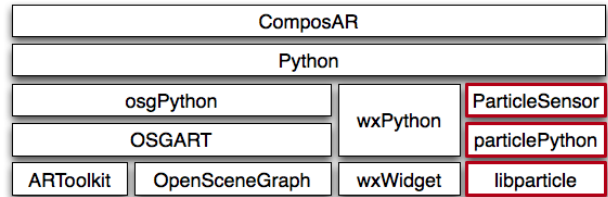


Figure 5. Sensor-based Interaction Module

of graphic rendering and scene graph management). In addition, osgART [6] is used to allow ComposAR to load video, have AR tracking functionality, and AR scene rendering.

Using the modified ComposAR application, we are able to support sensor-based script interactions in addition to marker-based script interactions. For example, the user can write interaction scripts based on the variation of sensory data in the following way:

```
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 the `__update__` function in ComposAR to allow a sensor to be used as a parameter as well as a marker. Thus, users can now interact with the virtual content by moving the sensor, changing lighting condition, or making noise. In this case changing the light value of the sensor will cause the virtual model to rotate in the AR scene. Figure 6 shows how covering the sensor with a hand causes the virtual model to rotate. By using our extension to composAR we can write many interesting context based interactions with the loaded virtual content.

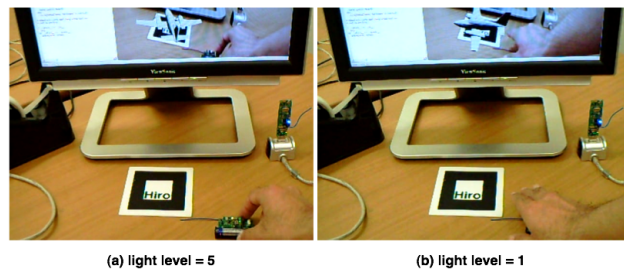


Figure 6. Interaction with the light level

ComposAR also allows users to save the current scene data and their interaction scripts in a XML file format for later retrieval. However, the current version of ComposAR did not support the ability to embed interaction scripts into

the XML file format. Thus, we also modified ComposAR to allow python interaction scripts to be included in the XML file, as shown in the following XML code.

```
<?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>
```

4. Discussion and Future work

In this paper, we have described a sensor-based interaction tool for U-VR based on an extension to the existing ComposAR script-based authoring tool for AR systems. ComposAR allows users to easily build and test simple AR systems due to scripting environment. With our modifications users can write their own interaction scripts that allow the values of sensor data to be used to modify virtual object properties. 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. In this way, the users are able to selectively share their contents as well as interactions with others. By making these extensions we have created a tool that allows users to rapidly author simple U-VR applications.

This is early work and in the future we would like to add more modules to create a complete U-VR authoring environment. For example we would like to allow the users to author interactions not only with fiducial markers but also more natural objects. We need to investigate the provisions of useful interaction template scripts for ordinary users, which use sensor based interaction metaphors rather than fiducial markers. Finally we will also need to evaluate the usability of the interface for sensor-based interaction to make sure it is as intuitive as possible.

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.