iOrb - Unifying Command and 3D Input for Mobile Augmented Reality

Gerhard Reitmayr, Chris Chiu, Alexander Kusternig, Michael Kusternig, Hannes Witzmann
Vienna University of Technology
reitmayr@ims.tuwien.ac.at,{e9825608|e0026571|e9926002|e0126743}@student.tuwien.ac.at

Abstract

Input for mobile augmented reality systems is notoriously difficult. Three dimensional visualization would be ideally accompanied with 3D interaction, but accurate tracking technology usually relies on fixed infrastructure and is not suitable for mobile use. Command input is simpler but usually tied to devices that are not suitable for 3D interaction and therefore require an additional mapping. The standard mapping is to use image plane techniques relative to the user's view. We present a new concept - the iOrb *- that combines simple 3D interaction for selection and dragging with a 2D analog input channel suitable for command input in mobile applications.*

Keywords: Command Input, 3D Interaction, Mobile Computing, Augmented Reality

1. Introduction and related work

Three dimensional input for mobile augmented reality (AR) systems is usually hard, because accurate tracking technology relies on fixed infrastructure and bulky hardware. As a result the standard tracking devices used in such setups are source-less inertial trackers for orientation of the head, cameras for optical inside-out tracking and GPS. Usually only the head of the user is tracked in full 6D pose because more devices are costly and require additional processing effort.

As a result 3D interaction is usually limited to the user's view and makes heavy use of image plane techniques which map a 2D cursor in the image plane to a constraint 3D interaction such as translation, rotation or scale. The same 2D cursor can also be used for ray intersection to allow selection and other forms of manipulation at a distance. A single system uses two 2D cursors to allow more complex manipulations [6]. Other approaches use optical 3D inside-out tracking by reusing the camera that is used to capture the video background for video see-through AR. Such solutions may depend on fiducials [7] or use natural feature tracking [3], but usually are not very accurate and robust.

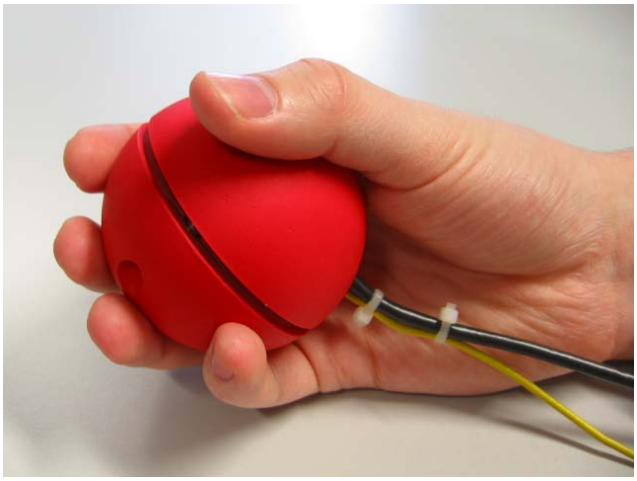

Figure 1. The *iOrb* operated by a user.

Command input in mobile AR applications is better developed. A variety of devices can be used to manipulate menus and widgets in a heads-up display. For example, hand-held trackballs [2], touchpads worn by the user [8], pinch gloves [5] or simple buttons are used in several systems. Another possibility is the use of an additional hand-held device such as a PDA or tablet PC [2] for both display and interaction with a 2D GUI that controls the application. However, such input devices are again only useable as 3D input devices via image plane techniques.

We present a new interaction device dubbed *iOrb* - interactive orb - that unifies the common ray casting operation with command input in a single device that can be used intuitively without image plane techniques and therefore decouples the 3D operation from head movements. The iOrb consists of an off-the-shelf inertial tracker for measuring 3D orientation build into a convenient sphere-like case and a single button, activated by pressing the two half-spheres together (see Figure 1).

The resulting device overcomes the limitation of 3D interaction being restricted to the user's view. The measured orientation can be used together with the head position to create a second 6D pose that is partly independent from the view pose. The device pose is used for ray casting to al-

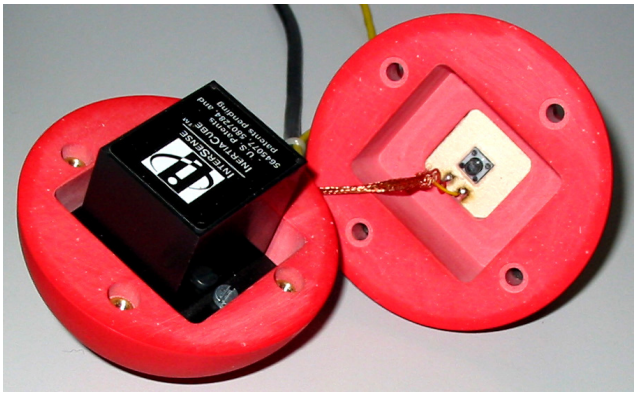

Figure 2. The interior houses a commercial inertial tracker measuring 3D orientation and a simple switch triggered by pushing the two hemispheres together.

low view independent selection and interaction which can be beneficial for dragging and other manipulations.

Moreover, the iOrb can be reused as a 2D input device similar to a trackball. The 2D input is achieved by mapping the 3D orientation to a 2D parametrization. Together with a build in button a user can operate hierarchical menus and other widgets like sets of checkboxes or lists.

2. iOrb hardware

Two hemispheres form the iOrb case providing room for an orientation tracker and a small button, each fixed in one half. The two parts are connected with screws and kept apart with coil springs. The mechanics enable the user to activate the button by pressing the ball which does not prescribe a certain way of holding the device to activate an external button. The iOrb is easy to dismantle without destroying any parts. Hence, the tracker can simple be exchanged with other models.

3. Principles of operation

To use the iOrb as a 2D input device we need to define a 2D parametrization of the 3D orientation. Then the resulting two dimensional parameter vector can be used as input. One way to define a mapping of a 3D orientation to a 2D analog device is to decompose the full 3D rotation into two consecutive rotations around orthogonal axes. The two angles become the two input dimensions.

However, the choice of axes is crucial. Here the main idea is to sense a main axis from the current orientation of the user's hand instead of using a fixed axis. The rotation axis for supination/pronation of the forearm (see Figure 3) is sensed during the first moments of the interaction

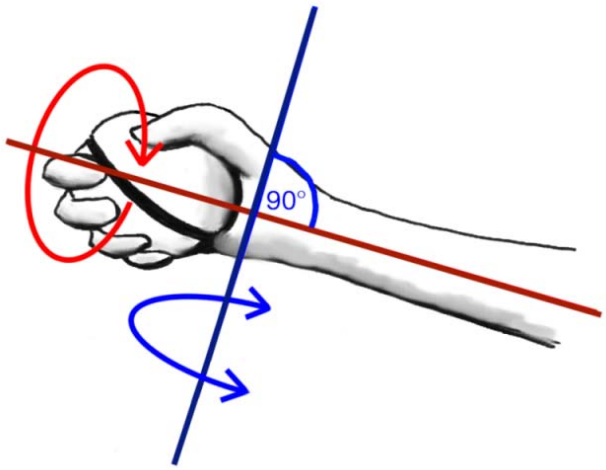

Figure 3. Forearm supination and pronation define the main rotation axis of the iOrb (red). The secondary rotation axis (blue) is orthogonal to the main axis.

to determine such a main axis. The assumption behind this choice is that such a movement is the simplest for the human hand and will therefore be used almost unconsciously and as the first intuitive movement of any interaction. Other movements induced by wrist flexion/extension or abduction/adduction require more awareness of the user.

3.1. Deriving two analog dimensions

Any interaction with the device begins with an explicit action by the user. The program stores the initial orientation of the device as reference to which all further movements are relative to. In the next initialization state it samples the first measurements to establish a main axis. In the final operational state the two input values are computed continuously and drive widgets. The operational state ends with the end of the interaction. The detailed computations are explained in the following.

During the initial state the software samples n measurements and computes the optimal fitting rotation axis for the rotations relative to the reference orientation. The number of measurements is small ($n = 10$) and therefore the time interval is very short (100 ms) and does not disturb the interaction. The rotation axes are transformed to lie in the same half-space by comparing them to the first sampled rotation axis and mirroring them as necessary. Then we simply average all the rotation axes and normalize the result to compute an approximating main axis. A more accurate computation would minimize the angle between the optimal axis and all measured axes while enforcing the constraint of the axis be-

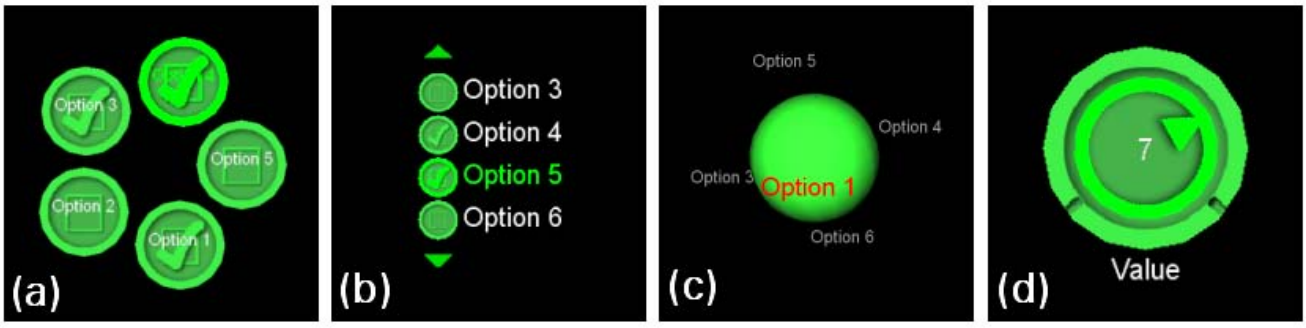

Figure 4. The set of available widgets for the iOrb.(a) Rings widgets (b) Array or linear widgets (c) The Option Ball maps the 3D rotation to a selection (d) Fader widget for continuous values.

ing of length 1. However, the difference is negligible in the standard operation.

In the operational state the sensed rotations r_s are decomposed into two factor rotations r_m, r_o such that r_m is a rotation around the main axis, r_o is a rotation around an axis orthogonal to the main axis and $r_s = r_o \cdot r_m$. Refer to [4, chapter 8] for a detailed analysis of such factorizations. During the interaction we get the following output from the process: the fixed main axis, the angle of rotation around the main axis, the variable secondary axis, and the angle of rotation around the secondary axis.

3.2. Ray casting

A number of possibilities exist to derive the direction for ray casting from the device. In the simplest form its absolute orientation can be used. However, this approach would require calibration at system start and during operation, because inertial trackers do not operate with a fixed reference frame and accumulate errors during operation. Another approach is to only use the relative orientation to the pose at the start of the interaction. Finally we can also map the 2D input derived in the last section to a fixed rotation scheme. For example, we can map the main rotation to the azimuth of the resulting direction and the secondary rotation to the elevation. The result is a standard way of using the ray casting interaction independently of the device's pose.

4. Widgets

In order to optimize the iOrb's method of interaction, user interface widgets have to be adapted. Here we followed traditional 1D command widgets as used before in virtual environments [1, chapter 8]. We created widget types that are both familiar in style, yet are intuitively usable through a rotational parameter. This parameter is typically the primary axis rotation provided by the 2D parametrization mentioned above. A second dimension for interaction can also be added by utilizing the secondary axis rotation.

4.1. Menu System

Part of the iOrb concept is a hierarchical menu system concept including a set of widgets optimized for the operation of the iOrb. Utilizing these widgets, the typical command input tasks of applications are possible. The common classes of widgets in this framework are ring widgets, array widgets, a fader-style widget, and a full rotational command selection widget (the Option Ball).

The hierarchy is obtained by implementing a system where the user can go forward to the next menu by clicking a menu item in a command array or command ring, and a mechanism to return to the previous menu. Two methods of "menu return" were designed: a button press time out mechanism and utilizing the secondary rotation from the parametrization. In the first method, pressing the button for a specified time interval will return the user to the previous menu. In the latter method, the secondary rotation crossing a specified threshold angle triggers the return.

Ring Widgets An intuitive way to map a single rotation to user interface interaction is to reflect this rotation directly. The result is the ring widget, a list of menu items or option items arranged in circular form (see Figure 4a). The primary rotation given from the 2D parametrization is mapped to the ring's rotation, either directly or with a certain rotation scale factor. Selection of a menu item, command item, or option is done by changing the orientation of the iOrb, and a click on the iOrb button will activate the command or check the option. Ring widgets can incorporate a list of commands or a single or multi selection option list.

Array Widgets Array widgets are list type widgets that display their content in a linear form. Here the primary rotation from the 2D parametrization is mapped to a linear

movement of the list and the selection focus (see Figure 4b). This widget supports scrolling of list items in order to support long lists. The iOrb click has the same semantics as with ring widgets. Array widgets can incorporate the same kinds of lists that ring widgets can - command lists, and single or multi selection option lists.

Fader Widget The fader widget was modeled after the commonly used real life widget, the fader control (sometimes called knob or dial control). This widget is capable of dialing a value within a certain range. The primary rotation of the parametrization directly influences the position of the dial, which in turn reflects a new value (see Figure 4d).

Option Ball The option ball doesn't utilize the 2D parametrization but maps the rotation directly. Option elements are arranged around a ball, which rotates according to iOrb input (see Figure 4c). The element in front is the selected one. The selection is achieved by a picking mechanism; the option ball contains a logical geometric element with a certain number of faces, which are picked to determine the selected element. Thus, if this logical geometric element is a cube, 6 option elements are possible.

4.2. Discrete picking

With the combination of a position and the orientation from the iOrb device, a variation on ray casting is possible. The position will typically be obtained from a positional tracker, e.g. GPS coordinates, and act as the source point for a ray casting action.

The orientation then determines the direction of the ray. A 3D point set is provided by the framework user out of which the picker will select one or none (discrete picking). The selected point is calculated by comparing the angle between the ray vector and the vector between the start position and the point in question. An additional separation angle can be specified by the user to limit the selection process to a certain tolerance level. A high separation angle will typically create a picker setup that will always select one point, while a low separation angle will make it necessary to point more accurately. This widget is not a widget in the sense that it visualizes itself; it can be visualized and integrated into a setup as the application programmer sees fit.

5. Conclusions and future work

The work is currently in a very early stage. We have assembled the device and implemented the basic software to derive the 2D input and a set of widgets for command input and ray casting. As a next step we will include it into a mobile AR system and use it as the sole input device for the system replacing a touch pad currently in use. We would like to experiment with the possibilities of operating the ray casting direction independently from the view direction. Selection and simple manipulation of objects will be the focus of the 3D interaction.

We also require user studies to see if and what differences are there in manipulating the widgets with either a trackball, a touchpad or the iOrb. Another direction is to compare the different approaches to controlling the ray casting direction with the device. Finally, symbolic input for text is interesting, for example using the Dasher [9] input method.

Acknowledgments

This work was sponsored by the Austrian Science Foundation *FWF* under contracts START Y193, and Vienna University of Technology by an infrastructure lab grants ("MARDIS").

References

[1] D. A. Bowman, E. Kruijff, J. J. LaViola Jr., and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison-Wesley, 1 edition, July 2004.

[2] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computer & Graphics*, 23(6):779–785, 1999.

[3] M. Kölsch, M. Turk, and T. Höllerer. Vision-based interfaces for mobility. In *Intl. Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, August 2004.

[4] J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, 1998.

[5] W. Piekarski and B. H. Thomas. Tinmith-hand: Unified user interface technology for mobile outdoor augmented reality and indoor virtual reality. In *Proc. IEEE VR 2002*, Orlando, Florida, USA, March 2002. IEEE.

[6] W. Piekarski and B. H. Thomas. Augmented reality working planes: A foundation for action and construction at a distance. In *Proc. ISMAR 2004*, Arlington, VA, USA, November 2–5 2004. IEEE.

[7] G. Reitmayr and D. Schmalstieg. OpenTracker – an open software architecture for reconfigurable tracking based on XML. In *Proc. IEEE Virtual Reality 2001*, pages 285–286, Yokohama, Japan, March 13–17 2001.

[8] B. H. Thomas, K. Grimmer, D. Makovec, J. Zucco, and B. Gunther. Determination of placement of a body-attached mouse as a pointing input device for wearable computers. In *Proc. ISWC'99*, pages 193–194, San Francisco, CA, USA, October 18–19 1999. IEEE.

[9] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay. Dasher - a data entry interface using continuous gestures and language models. In *Proc. UIST 2000*, pages 129–137, San Diego, CA, USA, November 5–8 2000. ACM.