Making Tracking Technology Accessible in a Rapid Prototyping Environment

Maribeth Gandy, Blair MacIntyre, Steven Dow
Interactive Media Technology Center, GVU Center, College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
{maribeth, blair, steven}@cc.gatech.edu

Abstract

In this paper we present an approach for exposing tracking technology in an accessible and flexible way to users of a rapid prototyping system for mixed (MR) and augmented reality (AR). Our system provides a tracking framework that alleviates the need for a high level of expertise while also presenting a model of the technology that allows for flexible modification of tracking configurations, the ability to quickly change an application from one type of tracking technology to another, and the creation of synthetic trackers for playback of prerecorded data, data fusion from multiple trackers, and wizard-of-oz applications.

1. Introduction

We have created the Designer's Augmented Reality Toolkit (DART) with the goal of providing a system that allows high level designers rather than technologists to work with AR experiences from initial prototyping, experience testing, and through deployment [1]. The DART system is built on top of Macromedia Director and consists of a low-level C++ plugin (Xtra) that provides services such as VRPN[2], marker tracking, and video capture, as well as a suite of behavior scripts written in the Director programming language (Lingo) that represent the components of an MR application such as trackers, 3D models, cameras, and events. The goal was to create a set of MR specific components that could be used by the developer following the familiar Director paradigm.

An important element of any MR application is tracking. To register graphical objects with the real world it is necessary to know the position and orientation of the user; interactive applications may need to be aware of the user's hands, other users, or the location of physical objects. One obstacle that prevents designers from creating MR applications is the expertise and domain knowledge required to work with tracking technologies and the complexities intrinsic to working with such hardware. Tracking technology is often difficult to work with for a variety of reasons. It can be expensive and difficult to configure. Interfacing between the tracker and your computer application can be complicated and require low level programming expertise. The requirement of tracking means that it is difficult to develop your application off-line without the hardware available, and it can be difficult to change which type of tracking technology your application uses. In DART we have implemented a new approach to tracker management and leveraged existing technology such as VRPN to alleviate many of these problems. The result is a flexible approach to tracking that allows for experimentation, off-line development, wizard-of-oz testing, and easy migration from one tracking technology to another.

2. Related Work

There are other tools and libraries that have been created to aid in the development of MR and VR applications that have taken different approaches to tracker management. For example, StudierStube [3] is an API intended for technically savvy users that provides tracker access via OpenTracker [4], middleware which sits between heterogeneous tracking hardware and the API. StudierStube provides a powerful API for tracker interaction, that requires extensive knowledge of trackers, object oriented programming, and Open Inventor. For example, a StudierStube developer could relatively easily modify and switch between trackers but in DART these actions could be accomplished by changing a property page setting or moving a component on the score. Also, the DART tracking model can be leveraged to create synthetic trackers that replay or fuse data.

3. The Tracking Architecture

3.1. Low Level Configuration

DART-Framework is a set of behaviors related to basic low level components of an MR application. Included in this collection are the behaviors *VRPN* and

MarkerTracking. A developer places these on the score and edits their property pages to set global configuration values and to activate these types of tracking. Also in this set of behaviors is a global script which serves to receive and transmit tracker reports to subscribers. VRPN and marker tracking can be used simultaneously and there is no DART imposed limit on how many trackers a developer can have active at a time.

3.2. Subscribers

DART-Actors is the collection of behaviors that represent objects (actors) in a MR experience. In the property page of an actor, the developer defines a tracker hierarchy making one actor a child of another, setting a local transformation, and linking the actor's position, rotation, or both to a tracker. Tracker specific functions can be applied on a per actor basis; allowing for different handling of a single tracker by each actor. This tracker can be either a real-life one such as an ARToolkit fiducial, or it can be a synthetic tracker using pre-recorded data; the nature of the tracker is transparent to the subscribers thus making it easy to switch between types of tracking as well as between real and synthetic trackers.

3.3. Synthetic Trackers

The DART tracking infrastructure enables the creation of synthetic trackers due to the routing of tracking data through a central location, the subscription model, and the use of uniform timestamps for all tracking reports.

To capture tracker data a developer simply places a *CaptureTracker* behavior on the score for each tracker she wishes to record. These behaviors become subscribers to tracker data just like the actors; however, when the data is received it is saved to a file for later use. To playback tracker data a *PlaybackTracker* behavior is placed on the score. The developer defines via the property page which saved data should be used, and what name this new synthetic tracker should broadcast as. Playback data can be used simultaneously with live trackers.

Similar to how the playback tracker works, the developer can also choose to fuse the data from two different trackers and then broadcast them as a new tracker. A *FusionTracker* behavior is placed on the score and the developer defines which two trackers will feed into it and which values the trackers will effect (position, rotation, or both). One of the trackers is marked as being "ground truth." If one tracker provides position and another orientation then there is no overlap and the reports are simply fused together and broadcast. If they overlap, the ground truth value is used when available, but when a value from the second tracker comes in the delta change from the last ground truth report is used to calculate a new report that is then broadcast. For example, an inexpensive tracking solution could be crafted from fiducials with their world transform defined and an inertial sensor to provide rotation values when no fiducials are in view.

This concept of synthetic trackers also makes it possible to replace a real tracker with a wizard-of-oz implementation for early experimentation and testing. For example, in an audio experience we replaced actual position tracking with a software application showing a map of the space, the operator would watch the user's movement and use the mouse to indicate her position in the space. This position information was then fed into the application as actual tracking data and fused with orientation values from an inertial sensor.

4. Conclusion

The DART tracking framework provides a flexible and easy to use interface between tracking technology and MR applications. This approach to the representation and functionality of trackers is not specific to DART and could be implemented under other MR and VR toolkits with the hope that by reducing the obstacles associated with tracker usage we could make MR application development accessible to designers, HCI researchers, students, and others who can advance the exploration of this medium.

5. References

[1] B. MacIntyre, M. Gandy, J. Bolter, S. Dow, B. Hannigan, "DART: The Designer's Augmented Reality Toolkit", *Extended abstracts of the International Symposium on Mixed and Augmented Reality (ISMAR'03) and the conference on User Interface Software and Technology* (UIST'03), 2003.

[2] R. Taylor, T. Hudson, A. Seeger, H. Weber, J. Juliano, A. Helser, "VRPN – A Device-Independent, Network-Transparant VR Peripheral System", *Virtual Reality Software and Technology* (VRST'01), Banff, Canada, Nov. 15-17, 2001.

[3] Zs. Szalavari, D. Schmalstieg, A. Fuhrmann, M. Gervautz, "Studierstube- An Environment for Collaboration in Augmented Reality" *Virtual Reality - Systems, Development and Applications*, Vol. 3, No. 1, pp. 37- 49, Springer, 1998.

[4] D. Schmalstieg, G. Reitmayer, "An Open Software Architecture for Virtual Reality Interaction", *Proceedings of ACM Symposium on Virtual Reality Software & Technology* (VRST 2001), 2001