TAMGeF: Touch-midAir-Motion Framework for Spatial Input

Francisco R. Ortega
Florida International Univeristy
fortega@fiu.edu

Naphtali D. Rishe
Florida International Univeristy
ndr@acm.org

Armando Barreto
Florida International Univeristy
barretoa@fiu.edu

CCS Concepts

•**Human-centered computing** → **User interface toolkits;** *Touch screens;*

Keywords

Input Devices; Framework; User Interfaces

1. INTRODUCTION

With the explosion of modern input devices, such as multi-touch, vision-based systems (mid-air interaction with devices such as Microsoft Kinect), and inertial navigational systems, the complexity of development has increased. There are different libraries or toolkits that support different features. For example, VRPN[1], which is a C/C++ (primarly C-based) library with socket client/server architecture, provides access to different multiple input devices. Other framework or libraries exist, such as OpenNI that provide vision-based functionality. However, VRPN is meant as the layer between input devices and the developer. We propose a framework that goes further in providing a complete solution. Takala et al. provided a good understanding about the current development challenges for 3D User Interface (3DUI) developers [3]. TAMGeF is based on previous input taxonomies [2], multi-touch interaction framework research (e.g., Mudra and Midas) [1], and our experience accumlated over time. In this paper, we explore two contributions: device abstraction and parallel message handling.

2. TAMGeF FOR SPATIAL INPUT

TAMGeF, a modern C++ cross-platform, template-based, multi-threaded framework for spatial input devices, is designed with multiple layers to provide greater flexibility. The primary layers are: (0) input devices; (1) platform core (desktop, web, mobile); (2) gesture recognition; (3) Toolkit modules (plugin, experimentation, task-parallelism, and visualziation); (4) bindings (e.g, Python, Unity, Java, etc.). In this paper, we will discuss part of the lower layer.

We have modified the definition from [2] to a generalized input device: (1) manipulation (M) provides the degrees of freedom (DOF) that a device supports. For example, a 6-DOF can be defined as $M = \{Tx, Ty, Tz, Rx, Ry, Rz\}$. (2)

[1]http://www.cs.unc.edu/Research/vrpn/

SUI '15 August 8–9, 2015, Los Angeles, CA, USA
© 2015 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.1145/2788940.2794355

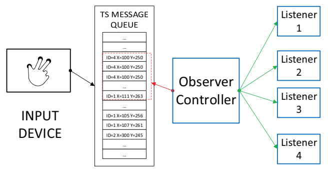

Figure 1: Observer-Queue Controller

Input Domain (In) provides the original device domain. For example, a multi-touch display may provide X and Y values between 0 to the maximum resolution. (3) The resolution function (Rf) provides a way to map the input to the output domain. (4) The output domain (Out) is the desired result provided by *Rf*. (5) The possible states (S) of the device (e.g, down, move, up, idle, cancel, for multi-touch). (6) The possible events Ev that the device may fire. This is related to S but it provides the mechanism to notify listeners. (7) The connection (Cn) provides the concrete mechanism to connect the device D. Therefore, a device is defined by a 7-tuple: ***D* = <*M, In, Rf, Out, S, Ev, Cn*>**.

There are multiple ways to fire an event: callback function, message queue, and observer pattern, among others. They all have drawbacks. In particular, the observer pattern creates problems when working in a multi-threaded environment, as we have experienced and it has been noted by others [1]. While TAMGeF does provide the flexibility to choose the mechanism, we provide some default behaviours. We have proposed a specialized pattern for input devices called **Observer-Queue Controller**, shown in Figure 1. This is an approach to combine a thread-safe message queue, an observer controller, and a set of listeners. This reduces the amount of thread locks and provides a simpler interface.

3. CONCLUSION

This paper provided a brief understanding of the lower layer of **TAMGeF** providing the generalization of a device (which can use set-theory to describe it) and a highly-optimized, thread-safe observer-queue controller pattern. We are currently developing this solution at our institution.

4. REFERENCES

[1] L. Hoste, B. Dumas, and B. Signer. Mudra: a unified multimodal interaction framework. In *ICMI '11: Proceedings of the 13th international conference on multimodal interfaces.* ACM, Nov. 2011.

[2] J. Mackinlay, S. Card, and G. Robertson. A semantic analysis of the design space of input devices. *Human–Computer Interaction*, 5(2):145–190, 1990.

[3] T. M. Takala, P. Rauhamaa, and T. Takala. Survey of 3DUI applications and development challenges. *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 89–96, 2012.