TAMGeF: Touch-midAir-Motion Framework for Spatial Input

Francisco R. Ortega  Naphtali D. Rishe  Armando Barreto
Florida International University  Florida International University  Florida International University
fortega@fiu.edu  ndr@acm.org  barreto@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++ (primarily 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 accumulated 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 visualization); (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 \(M = \{T_x, T_y, T_z, R_x, R_y, R_z\}\). (2)

\(\text{INPUT DEVICE} \rightarrow \text{Listener} \rightarrow \text{Observer-Queue Controller}\)

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


---

\(^{1}\)http://www.cs.unc.edu/Research/vrpn/