MIDAS-M: A Software Framework for Supporting Multimodal Interaction on Heterogeneous Interaction Devices for Cloud Applications

Myunghee Lee[1], Gerard J. Kim[1](✉), and Jeonghyun Baek[2]

[1] Digital Experience Laboratory, Korea University, Seoul, Korea
{lmh81,gjkim}@korea.ac.kr
[2] KMC Robotics, Seoul, Korea
jhbaik@kmcrobot.com

Abstract. In this paper, we present a software framework, called MIDAS-M (Mixing and matching heterogeneous Interaction Devices to Applications and Services) that enables an application to lend itself to many different types of interaction methods and accommodate users with different client devices in a flexible manner. In particular, we focus on the aspect of supporting "multimodal" interaction by defining and mapping events that are mixed and matched by different input/output components. The multimodal events defined this way can be realized on various client platforms according to their capabilities. We also describe a case study of applying MIDAS-M to developing a multimodal interface for a virtual apartment preview system, called the SMARTIS, and demonstrate its advantages.

Keywords: Cloud based interaction · Ubiquitous interaction · Multimodal interaction · Software framework

1 Introduction

Cloud computing allows and simplifies the instant access of rich information and high quality service anywhere by decoupling the core software functionality (on the server side) and the input/output (on the client side). That is, the client can be relatively computationally less capable supporting just the acquisition of user input, presentation of the output and communication these between the client and server. In many cases, the web environment is used to provide the interaction platform independence for the cloud client. The web environment basically supports only the standard keyboard and cursor/pointer/touch interaction. In the future, we can easily imagine a situation where users with variant client devices (and interaction capabilities) wanting to accomplish a given application task in a different way. For instance, in playing a first person shooting game, one might prefer to shoot using a hardware button, another by touch button, another by swipe gesture, and another by voice command. That is, sophisticated users will expect to choose the most usable form of interaction for themselves, depending on the given device (e.g. desktop, smart phone, pad, VR).

N. Streitz and P. Markopoulos (Eds.): DAPI 2017, LNCS 10291, pp. 172–179, 2017.
DOI: 10.1007/978-3-319-58697-7_12

Traditionally, to accommodate the situation as described above, either separate client programs are developed for (or ported to) different devices and operating systems, enforcing a particular interface most suited for the given device, or a "large" client program is developed to cover all possible interaction possibilities. Practically, this has caused application services (cloud or local) to be compatible to only a small family or brand of devices and leaving no choices for the users in terms of the interaction possibilities.

In the line of such a need, we had developed present a software framework, called MIDAS (Mixing and matching heterogeneous Interaction Devices to Applications and Services), that enables an application to lend itself to many different types of interaction methods (namely, sensing and display) and accommodate users with different client devices in a flexible manner [1]. In this paper, we present an important extension to MIDAS, called MIDAS-M, for supporting "multimodal" interaction in the similar way. Using MIDAS-M, multimodal events can be defined in an abstract manner, and mapped to the application, e.g. to provide redundant and assured input/output delivery, alternative interaction methods, and/or interfaces matched in modalities by the task characteristics. The multimodal events defined this way can be realized on various client platforms according to their capabilities. We describe the MIDAS-M framework and present a case study of applying it to developing a multimodal interface for a virtual apartment preview system, called the SMARTIS.

2 Related Work

The objective of our work is similar to those of the migratory or plastic interfaces. Migratory interfaces are those that operate on changing operating platforms or interaction resources. Cameleon-RT [2] and BEACH [3] are examples of such software architecture/infrastructures in which multiple interaction devices could be managed and interact with the given application. Such middlewares enable a "generically" described interactive application to build their own "view" for a given physical interaction platform. Plasticity further requires the migrating interface to preserve the usability as much as possible by adaptation to the specific operating platform (e.g. the control panel lay out changed according to the size of the PDA) [4]. In these approaches, separate platform specific implementations and compilation processes are still needed, and they do not support the design of multimodal interaction in a flexible mix (different modalities) and match (to various devices) fashion.

TERESA [5] is an authoring tool for "One Model, Many Interfaces" [6] type of applications. With the help of TERESA, different user interface codes can be conveniently generated from abstract user interface descriptions. However, the output application still compiled for a specific platform with its interaction capabilities known ahead of time. The SEESCOA component system [7] uses abstract UI described in XML which is interpreted to compose user interfaces at run time. Mark up languages for the interactive web and cloud applications that describe UI objects, and interpreted and realized by different platform browsers, operate on similar principles [8, 9]. Gilroy et al. has developed a client middleware that adapts application's "presentation styles" (expressed in an XML format specification document) most suited for a given device at run time. This way, the menus, form-fills, dialogue GUIs are adjusted and presented according to the device capabilities and thus high usability can be expected [10]. Our work focuses on defining and managing multimodal interaction.

3 MIDAS-M Framework

Figure 1 shows the overall architecture of MIDAS-M. In the far left (and right), lie different input (and output) modules/platforms operating in different modalities or equipped with various sensors and displays. MIDAS-M takes the raw input signals from the input modules (which may be separate from where the application resides), and relays them to the application (possibly through the network) in the form of meaningful application "events." The "events" can be multimodal that combines the different input streams in different timings (but within a limited time window) and/or contributing factors. For example, a multimodal input might be defined as a successive input of a voice command and finger tracking data within 1 s, or a simultaneous input of voice and touch. Such definitions are specified by the developer using a separate authoring tool (Fig. 2) producing an XML based description of the multimodal events (Fig. 3), as an input to the MIDAS-M engine (middle part of Fig. 1). As illustrated in Fig. 3, the events

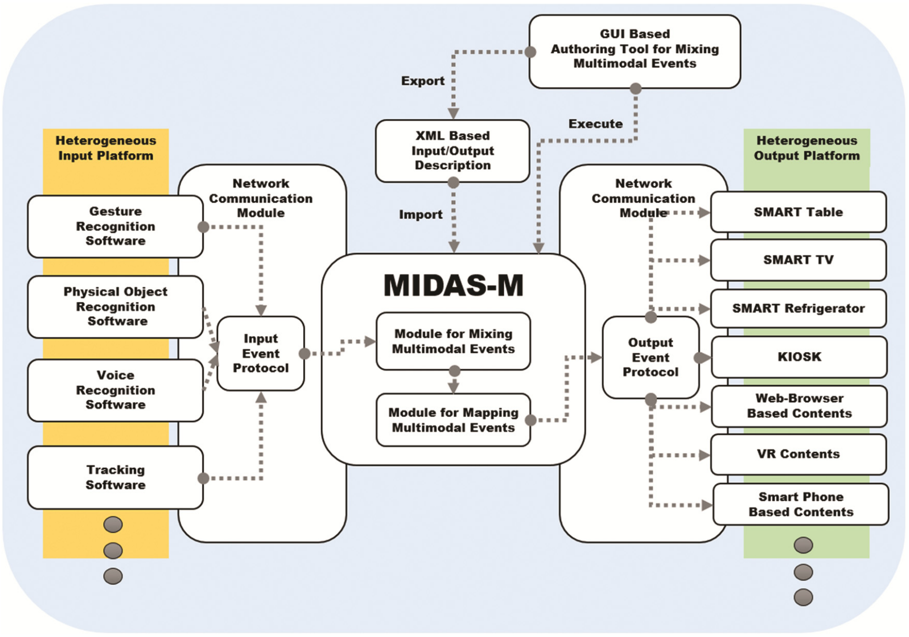
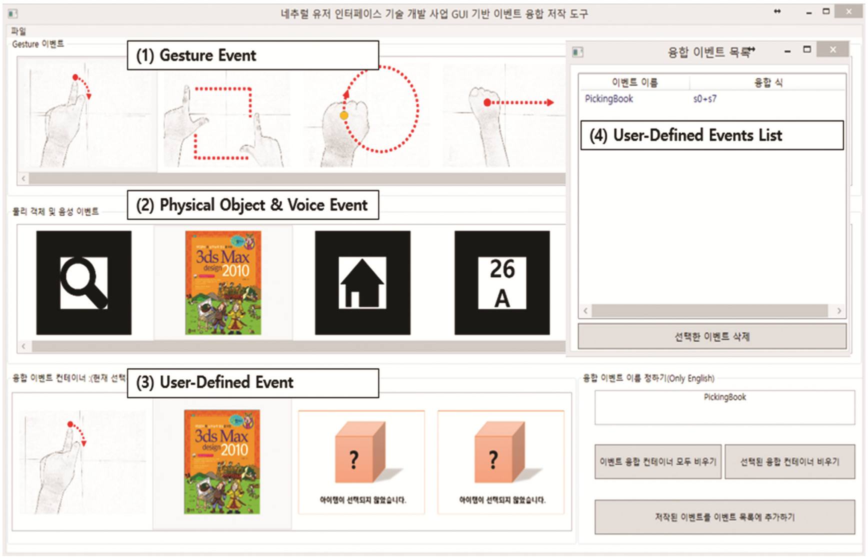

GUI Based Authoring Tool for Mixing Multimodal Events
Export
Execute
XML Based Input/Output Description
Import
Heterogeneous Input Platform
Gesture Recognition Software
Physical Object Recognition Software
Voice Recognition Software
Tracking Software
Network Communication Module
Input Event Protocol
MIDAS-M
Module for Mixing Multimodal Events
Module for Mapping Multimodal Events
Network Communication Module
Output Event Protocol
Heterogeneous Output Platform
SMART Table
SMART TV
SMART Refrigerator
KIOSK
Web-Browser Based Contents
VR Contents
Smart Phone Based Contents

Fig. 1. MIDAS-M: A flexible architecture that decouples the application from interaction complexities and device heterogeneity. It also supports definitions of multimodal input and output events as mapped to the application in a neutral and generic way.

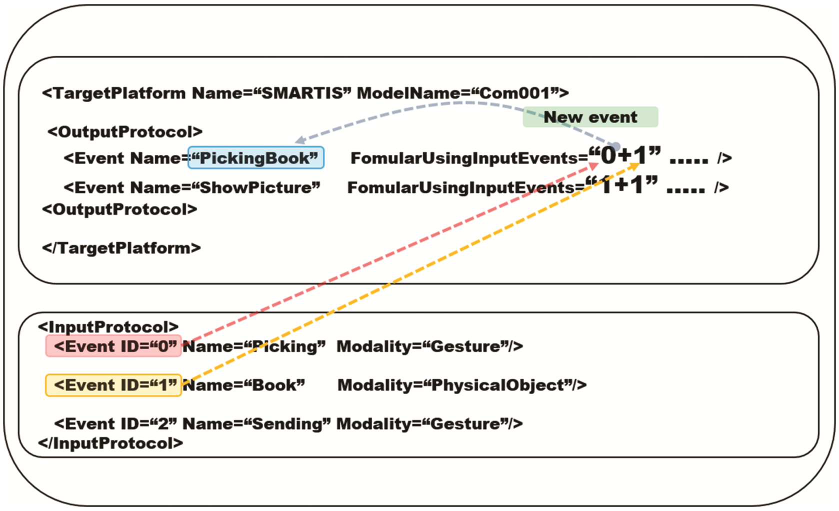
are assigned with unique identifiers and used in the given application to define their handlers.


네추럴 유저 인터페이스 기술 개발 사업 GUI 기반 이벤트 융합 저작 도구
파일
Gesture 이벤트
(1) Gesture Event
물리 객체 및 음성 이벤트
(2) Physical Object & Voice Event
3ds Max design 2010
26 A
융합 이벤트 컨테이너 :(현재 선택
(3) User-Defined Event
아이템이 선택되지 않았습니다.
아이템이 선택되지 않았습니다.
융합 이벤트 목록
(4) User-Defined Events List
선택한 이벤트 삭제
융합 이벤트 이름 정하기(Only English)
이벤트 융합 컨테이너 모두 비우기
선택된 융합 컨테이너 비우기
저작된 이벤트를 이벤트 목록에 추가하기

이벤트 이름	융합 식
PickingBook	s0+s7

PickingBook

Fig. 2. The GUI based authoring tool to define multimodal events and mapping them to a given application. An example of an XML description of a multimodal event



```
<TargetPlatform Name="SMARTIS" ModelName="Com001">
 <OutputProtocol>
   <Event Name="PickingBook"   FomularUsingInputEvents="0+1" ..... />
   <Event Name="ShowPicture"   FomularUsingInputEvents="1+1" ..... />
<OutputProtocol>
</TargetPlatform>

<InputProtocol>
  <Event ID="0" Name="Picking"  Modality="Gesture"/>
  <Event ID="1" Name="Book"     Modality="PhysicalObject"/>
  <Event ID="2" Name="Sending" Modality="Gesture"/>
</InputProtocol>
```

Fig. 3. An example of an XML description of a multimodal event. For instance, the output event, "PickingBook" is triggered by a combined (sequential) multimodal input event of "0" (hand gesture) and "1" (target object recognition).

Just like the multimodal input events, multimodal output events can be specified in the same way and used in the pertaining application. Important aspects of multimodal output event might include the timing, synchronization and heterogeneous device support and customization (e.g. according to display resolution, screen size, volume level, etc.).

4 A Case Application: SMARTIS

We demonstrate the flexibility of the MIDAS-M framework, in providing a variety of ways to interact and improving general usability, by applying it to the development of a virtual apartment previewing system, called the SMARTIS. It is composed of two main output displays, namely, a tabular and an upright monitors, and a number of sensors for input such as the Microsoft Kinect depth sensor, an RGB camera, touch screen, and a microphone.

With SMARTIS, the user can make commands in various ways for making selections (e.g. particular floor plan, apartment model/size, interaction/content modes, view-points), querying for information (e.g. price, location, options), zooming in/out, and navigating and exploring the scene. Many interactions are defined multimodally. For example, the "move forward" command (for navigating the virtual apartment) can be invoked by voice, touch or mid-air gestures separately or combined. Figure 4 shows the system in use and Table 1 summarizes the (multimodal) interfaces for various afore-mentioned tasks. The output is presented through the two monitors, e.g. the 2D floor plan on the flat monitor and 3D interior scenes on the upright, accompanied by sound/voice feedback where appropriate. Figure 4 shows the overall system set-up and its typical usage situation.

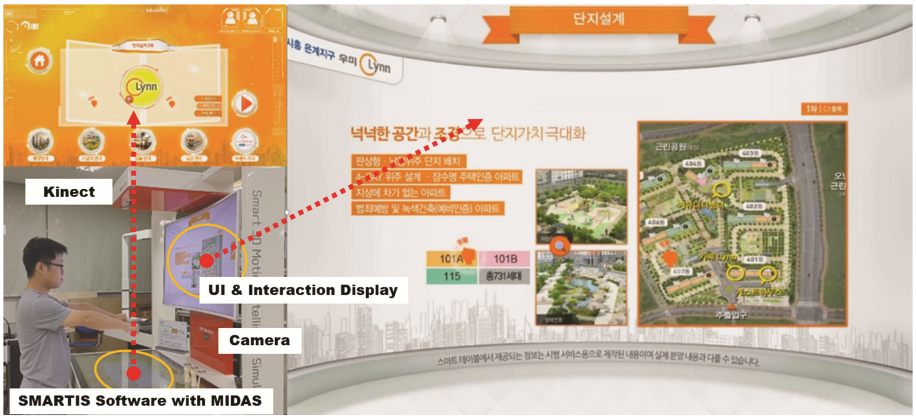

Kinect
UI & Interaction Display
Camera
SMARTIS Software with MIDAS
단지설계
넉넉한 공간과 조경으로 단지가치 극대화
101A
101B

Fig. 4. A scene from interacting with SMARTIS (virtual apartment previewing system) using natural gestures, finger tracking, touch, multiple displays and voice commands.

Table 1. Multimodal interfaces for various tasks in SMARTIS.

	Task	Interface
Input	Navigation (Translation and Rotation)	Gesture
		Voice
		Touch
		Object recognition
	Select	Object recognition
		Gesture
	Zoom in/Zoom out	Gesture
	Capturing ROI	Gesture
	Mode change	Voice
		Gesture
		Touch
Output	Present 2D floor plan	Visual feedback (flat display)
		Sound feedback
	Present 3D apartment interior	Visual feedback (upright display)
		Sound feedback
	Present instructions	Visual feedback (flat/upright display)
		Sound feedback
	Command confirmation	Visual feedback (flat/upright display)
		Sound feedback

The input modules and the core applications in SMARTIS were developed separately on different operating platforms and run on separate computers communicating through the network. The whole system was internally integrated through the event language and protocol of the MIDAS-M, which defined the various input and output events (some of multimodal, see Table 1) and mapped to the core application logic. These event specifications were mostly carried out using the aforementioned authoring tool (Fig. 2).

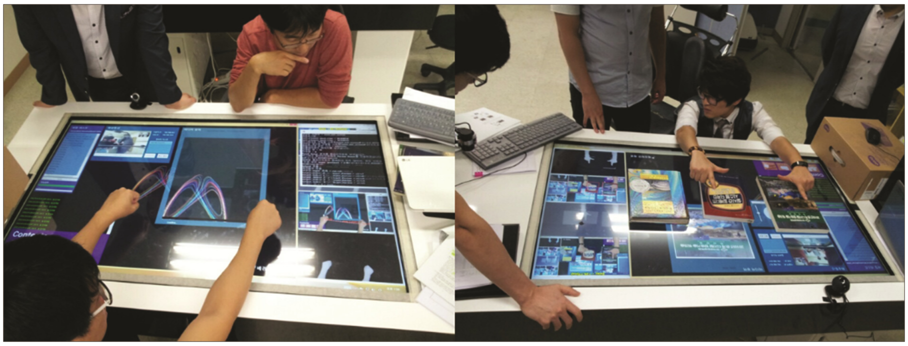


Fig. 5. Another smart table based object manipulation application quickly developed using the same I/O modules and MIDAS-M framework.

The decoupled development, between the I/O and core application follows the well-known principle of separation of concerns [11] and the MVC (Model-View-Controller) development methodology [12, 13]. All the input/output modules are easily reusable in different multimodal combination for another application. Figure 5 shows a simple smart table based object manipulation application, quickly developed using same I/O modules and MIDAS-M middleware.

5 Discussion and Conclusion

In this paper, we presented MIDAS-M that supports multimodal interaction and interface-model decoupled application development. The multimodal events can be defined in a variety of forms to allow rich interaction and high usability of the target application. The framework was put to use to develop a commercially deployed product. It demonstrates that how MIDAS-M allows a flexible mixing and matching of and reusing of the I/O modules at compile or even run time to suit the various interactional needs. In addition to the logical decoupling between the I/O and core application, future MIDAS-M framework can be applied as a cloud middleware (e.g. for hardware/platform decoupling) in which clients only have to perform the necessary I/O functions with the sensors and input modules available to themselves.

Acknowledgement. The research was supported through the “Software Convergence Technology Development Program,” funded by the Ministry of Science, ICT and Future Planning (S1002-13-1005), and also through the “Basic Science Research Program,” funded by the Korean National Research Foundation (No. 2011-0030079).

References

1. Ahn, E., Lim, K., Kim, G.J.: MIDAS: a software framework for accommodating heterogeneous interaction devices for cloud applications. In: Streitz, N., Stephanidis, C. (eds.) DAPI 2013. LNCS, vol. 8028, pp. 339–348. Springer, Heidelberg (2013). doi: 10.1007/978-3-642-39351-8_37
2. Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G.: CAMELEON-RT: a software architecture reference model for distributed, migratable, and plastic user interfaces. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, James L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 291–302. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30473-9_28
3. Tandler, P.: Software infrastructure for ubiquitous computing environments: supporting synchronous collaboration with heterogeneous devices. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) UbiComp 2001. LNCS, vol. 2201, pp. 96–115. Springer, Heidelberg (2001). doi: 10.1007/3-540-45427-6_9
4. Calvary, G., Coutaz, J., Thevenin, D.: A unifying reference framework for the development of plastic user interfaces. In: Little, M.R., Nigay, L. (eds.) EHCI 2001. LNCS, vol. 2254, pp. 173–192. Springer, Heidelberg (2001). doi:10.1007/3-540-45348-2_17
5. Mori, G., Paterno, F., Santoro, C.: Design and development of multidevice user interfaces through multiple logical descriptions. IEEE Trans. Softw. Eng. **30**(8), 507–520 (2004). IEEE

6. Paterno, F., Santoro, C.: One model, many interfaces. In: Kolski, C., Vanderdonckt, J. (eds.) Computer-Aided Design of User interfaces III, vol. 3, pp. 143–154. Springer, Netherlands (2002)
7. Luyten, K., Vandervelpen, C., Coninx, K.: Migratable user interface descriptions in component-based development. In: Forbrig, P., Limbourg, Q., Vanderdonckt, J., Urban, B. (eds.) DSV-IS 2002. LNCS, vol. 2545, pp. 44–58. Springer, Heidelberg (2002). doi: 10.1007/3-540-36235-5_4
8. XHTML2 Working Group Home Page. http://www.w3.org/MarkUp
9. XAML Overview (WPF). http://msdn.microsoft.com/en-us/library/ms752059.aspx
10. Gilroy, S.W., Harrison, M.D.: Using interaction style to match the ubiquitous user interface to the device-to-hand. In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004. LNCS, vol. 3425, pp. 325–345. Springer, Heidelberg (2005). doi:10.1007/11431879_22
11. Suh, N.P.: The Principles of Design, vol. 990. Oxford University Press, New York
12. Patterns, D., Pattern, C.: Model-View-Controller (2003)
13. Leff, A., Rayfield, J. T.: Web-application development using the model/view/controller design pattern. In: Enterprise Distributed Object Computing Conference, pp. 118–127. IEEE (2001)