A Framework for Constructing Entertainment Contents using Flash and Wearable Sensors

Tsutomu Terada† and Kohei Tanaka‡

†Kobe University, ‡Osaka University
†1-1 Rokkodai, Nada Ward, Kobe, Hyogo 657-8501, Japan
‡1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
tsutomu@eedept.kobe-u.ac.jp,tanaka.kohei@ist.osaka-u.ac.jp

Abstract. Multimedia interactive contents that can be controlled by user's motion attract a great deal of attention especially in entertainment such as gesture-based games. A system that provides such interactive contents detects the human motions using several body-worn sensors. To develop such a system, the contents creator must have enough knowledge about various sensors. In addition, since sensors and contents are deeply associated in contents, it is difficult to change/add sensors for such contents. In this paper, we propose a framework that helps contents creators who do not have enough knowledge on sensors. In our framework, an interactive content is divided into two layers; sensor management layer and content layer. We confirmed that creators can create interactive contents easier with our framework.

Key words: Wearable Computing, Flash, Development Environments

1 Introduction

In recent years, the importance of intuitive human-computer interfaces has increased and increased. Especially, *Nintendo Wii* that is a video game console released in 2006 infuses new breath into the game market since we can play video games intuitively by using sensor-enabled game controller as a sword and a tennis racket. On the other hand, it is still difficult for contents creators who create contents using *Flash* or other animation tools to develop such contents since they usually do not have the knowledge to manage sensors and to recognize user's motion from raw sensor data. Moreover, once the contents creator made a content using sensors, it is difficult to change the sensors used in contents because the sensor management is deeply integrated to the contents. Therefore, there is a requirement on the support for contents creation by easier sensors managements.

Here, Flash[6] is so popular for providing interactive contents on web browsers. Flash contents have various advantages such as small file size with respect to rich contents, and support for various platforms (more than 99% people in mature market can play flash contents). Contents creators can implements interactivity on the contents using easy programming scripts called *ActionScript*. Thus, we

```
WHEN GPS_MOVE
IF CURRENT.Position == 'station'
THEN DO BROWSER_OPEN('URL')
```

Fig. 1. An example of ECA rule

propose a framework for flash creators to create interactive contents with sensors and actuators. Our framework provides several tools to hide the difficulty in sensor management and creators can create interactive contents by ActionsScript easily.

The remainder of this paper is organized as the follows. Section 2 describes our environmental assumptions and Section 3 presents related works. Section 4 describes the process of developping contents using our framework. We explain our framework in detail in Section 5 and discuss our framework in Section 6. Finally, we conclude this paper in Section 7.

2 Environmental assumption

The target users of our framework are flash contents creators that do not have enough knowledge how to use the sensor data while they can create flash contents without sensors. Recent days, there are many people that can create flash contents since flash becomes popular.

We employ *Wearable toolkit* that we developed to define/recognize user contexts. It consists of an event-driven rule processing engine and several plug-ins to implement context-aware applications easily. The rule in the toolkit is called ECA rule, which consists of *Event*, *Condition*, and *Action*. For example, when we want to develop the system that *displays the train schedule when user arrives at the station*, we define the reception of GPS data as *Event*, the comparison the current position with the position of station as *Condition*, and the navigation to web site as *Action*, as shown in Figure 1. Wearable toolkit has *Context Definition Tool* that users define user contexts easily by actually demonstrating the motion to be registered. The rule engine raises an event when it detects the similar motions to registered contexts. The Context Definition Tool supports various sensors such as 3-axis acceleration sensors, temperature sensors, a GPS, and RFID tag readers.

Figure 2 shows an example of our assumed content and user. The user interacts with the content by his motions in front of the screen. He receives feedbacks by the screen, sound, and some actuators.

3 Related works

There are many researches to develop services using various sensors and actuators. Phigets[3] and Teleo[2] are toolkits that consist of various sensors and actu-

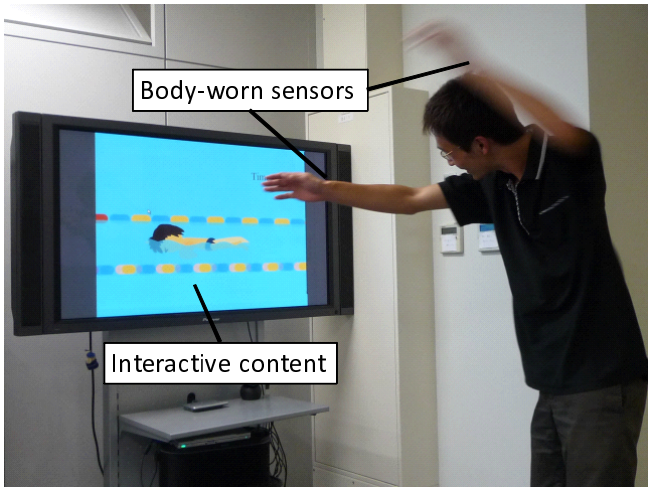


Fig. 2. An example of playing contents

ators to develop applications using the libraries in various development environments. Gainer[8] is a hardware platform to connect various devices to computer. GlovePIE[1] is a tool for assigning device functions to application functions by a script language. Using these toolkits, we can develop applications that use various devices easily. However, we still need to know how to process the sensor data since they support only connection between computer and devices.

A CAPpella[7] enables us to define user actions easily, and GT2K[9], ARToolkit[11], and DART[10] provide the recognition of gestures and markers by camera. However, these systems does not refer to programming model.

4 Contents development with our framework

We show an actual development of interactive game, a 110m hurdles game, as the following steps. Our framework especially supports Step 2, 3, and 4 since these steps are difficult for contents creators.

Step 1 Contents design
Step 2 Motion registratoin
Step 3 Rules description
Step 4 Flash contents creation
Step 5 Debug

Step 1 Contents design: We established the overall picture of the contents in this step. Then, We discussed the purpose of the contents, motions to be used, and required sensors and actuators. In this case, the purpose of the game is to compete the time from start to goal. The character on the flash contents runs while the player runs, it stops when the player stops, and it jumps when the

player jumps. Moreover, a vibrator works when the character fails to jump a hurdle.
Step 2 Motion registratoin: Next, we registered the player's motions that are designed in previous step using Context Definition Tool. To register them, we only performed the actual motions. We also confirmed that the actions can be recognized accurately. For this game, we registered the 3 motions; *run*, *stop*, and *jump*.
Step 3 Rules description: In this step, we described the rules for Wearable Toolkit to communicate with flash contents. Our *Code Generation Tool*, which is explained in the following section, produces the rules by inputting the relationship between user actions and flash scripts that are activated when the motions are recognized. Concretely, to create the rules about *run* action, we input the name of running action and a script that is executed at running. The Code Generation Tool generates the scripts and the rules to execute the input scripts when the system recognizes *run*. We also input descriptions about *stop* and *jump*. In the same way, we also easily get the scripts and the rules to control actuators by Code Generation Tool. In concrete, we selected the vibrating function from the list of Wearable Toolkit functions, then the named of the function can be used in flash contents.
Step 4 Flash contents creation: In this step, we created flash animation contents. The Code Generation Tool generated flash project file in which the script for communication is already described. To control an actuator, we only describe RaiseEvent() function that has the event name selected in the previous step. We added RaiseEvent() function in the script code for collision detection.
Step 5 Debug: We tested the contents using the debugger of Wearable Toolkit. Using this debugger, we confirmed that flash received the player motion events and Wearable Toolkit received the device control events. Moreover, we checked the accuracy in motion recognition. When the accuracy is not enough, we change the threshold in recognition or redefine the motions. We can change the content easily since Code Generation Tool generates rules and scripts independently from actions and actuators.

5 Design of framework

Our framework consists of 2 layers as shown in Figure 3. *Sensor management layer* manages sensors and actuators, and recognizes user motions. *Flash contents layer* is flash contents including several templates to communicate with the other layer. Since it is difficult for flash creator to use the sensor data directly, our framework hides the sensor management layer from contents creators. If they define the user motions that they want to use semi-automatically using *Context Definition Tool*, flash contents can receive the motion-recognition event from sensor management layer.

In this research, we add a function for communicating with flash contents to Wearable toolkit and implement *Code Generation Tool* that generates the rules and the scripts for associating motions with actions.

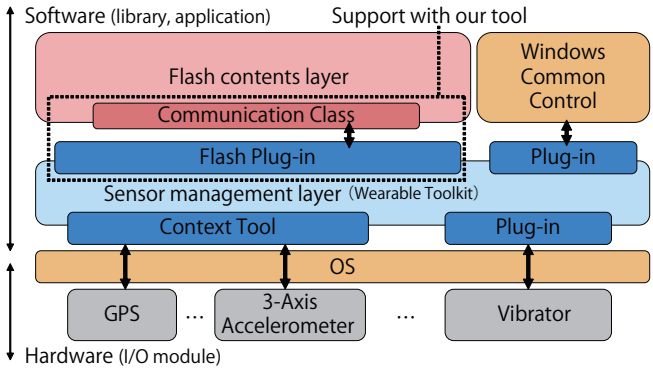


Fig. 3. Structure of proposed framework

```
RegisterAction(
    "ACTION_NAME", OnAction,
     "explanation", "type of argument");
RegisterEvent(
    "EVENT_NAME", "explanation",
     "type of argument");

function OnAction(Arg:type of argument){
 //OnAction implementation Here
}
```

Fig. 4. An example of script at initialization

To realize the communication between Wearable Toolkit and flash contents, we utilize the Flash Player's commands that are originally used for communication with Flash Player and the contents. Although this command cannot transfer without text message, it has enough capability to send context information to flash contents and we do not suppose the contents creators use the raw data from sensors. The communication between Wearable toolkit and flash occurs in the following three situations:

Communication on flash contents starting: First communication occurs in initializing the content. When a flash content starts, it sends messages to Wearable Toolkit to register user motions and actuators to be used via fscommand() function[4]. This process is based on the code shown in Figure 4, which is generated by Code Generation Tool. This process includes the association between motions and scripts. In the figure, `ACTION_NAME` action is associated with OnAction() function. When the toolkit receives the messages, it registers the messages as events or actions.

```
WHEN CONTEXT_RECOGNIZED('CONTEXT_NAME')
IF CURRENT.Rate > 80
THEN DO FLASH_ACTION('ACTION_NAME')
```

Fig. 5. An example of ECA rule to send a message to the flash content

```
WHEN FLASH_EVENT('EVENT_NAME')
THEN DO CONTOL_ACTUATOR()
```

Fig. 6. An example of ECA rule to control actuators

Communication when user motion is recognized: After the flash content starts, Wearable Toolkit recognizes user motions from sensor data. When a motion is recognized, ECA rules that are generated by Code Generation Tool are executed to send the recognition results to the flash content. The flash content calls the associated scripts according to the received results.

An example of such ECA rule is shown in Figure 5. This rule is activated when the context named `CONTEXT_NAME` is recognized, and sends the function name `ACTION_NAME` to exectute the script in the flash content. This rule is automatically generated by Code Generation Tool.

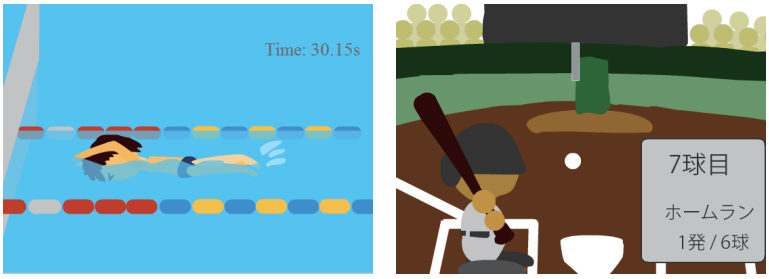
To realize this communication, `FLASH_ACTION` action calls SetVariable() function [5] on the SWFobject, which is a function to change the value of variables on the flash content. In the content, the target scripts are described as a callback function. In the example of Figure 4, `OnAction()` is executed when the value of the variable is changed to `ACTION_NAME`.

Communication when the content controls actuators: Our framework enables flash contents to control actuators by fscommand() function in the contents that sends text messages to Wearable toolkit. The ECA rules for actuators control in Wearable toolkit, which is created by our Code Generation Tool, receive commands and control the associated actuators. An example of the ECA rules is shown in Figure 6. The rule in the figure execute `CONTROL_ACTUATOR()` action to control an actuator when the command named `EVENT_NAME` is received via fscommand() from the flash content.

6 Consideration on implementation time

We discuss the easiness of contents development by using our framework. Firstly, we implemented an interactive game by modifying existing breakout game. We associated two motions on the game with two physical motions. We took less than one hour to modify this game.

Next, we implemented two games that are the homerun chase game and the swimming game. In the homerun chase game, to swing the bat on game, a user



(a) Swimming game (b) home run chase game

Fig. 7. Created games

swings an arm with an acceleration sensor. In the swimming game, when a user strokes with his arms with acceleration sensors, the game character swims forward. It took two hours for developing the homerun chase game and three hours for developing the swimming game. Most of the time for implementation was to create flash animations. Approximately 15 minutes are used for the definition of user motions and the description of rules that are needed to communications. Figure 7 shows snapshots of these games.

In addition, we created a content for donation box at Kobe Luminarie that is a festival held in Kobe, JAPAN. The festival is held to mourn the victims. We developed the content on the display in front of the donation box. The dog character in the content performs the same motion as a person who fold the donation box. For example, when the person bows, the dog on the display also bows and says, "Thank you for your donation" as shown in Figure 8.

This contents was also developed in approximately three hours. From these implementations, there is little time to implement interactive contents using sensors by our framework.

7 Conclusions

In this research, we proposed and implemented a framework that enables contents creators to create interactive contents using wearable sensors. The framework presents the mechanism for communication between Wearable toolkit and flash contents. Moreover, our proposed Code Generation Tool achieved automatic scripts/rules generation to use our framework easily. We confirmed that contents creator could develop intuitive contents in a few hours by evaluating actual implementations.

On the other hand, we feel that the difficulty of games becomes higher since the gesture input is difficult and delayed compared with the conventional input methods. Thus, we should adjust the game difficulty according to the recognition accuracy and the delay of the motion.

(a) waiting (b) bows

Fig. 8. Snapshots of the content in donation box

In the future, we will evaluate our framework based on actual implementations by flash content creators to show the effectiveness on complicated interactive contents.

References

1. Glovepie, `http://carl.kenner.googlepages.com/`.
2. Makingthings, `http://www.makingthings.com/`.
3. Phidgets, `http://www.phidgets.com/`.
4. Adobe: Actionscript 2.0 components lauguage reference, `http://livedocs.adobe.com/flash/9.0/main/`.
5. Adobe: Developer connection, `http://www.adobe.com/devnet/flash/articles/`.
6. Adobe: Flash CS4, `http://www.adobe.com/products/flash/`.
7. A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu: a Cappella: Programming by demonstration of context-aware applications, *Proc. of International Conference on Human Factors in Computing Systems (CHI2004)*, pp. 33–40 (2004).
8. S. Kobayashi, T. Endo, K. Harada, and S. Oishi: Gainer: A reconfigurable I/O module and software libraries for education, *Proc. of the 2006 International Conference on New Interfaces for Musical Expression (NIME06)*, pp. 346–351 (2006).
9. T. Westeyn, H. Brashear, A. Atrash, and T. Starner: Georgia Tech Gesture toolkit: Supporting Experiments in Gesture Recognition, *Proc. of International Conference on Perceptive and Multimodal User Interfaces(ICMI2003)*, pp. 85–92 (2003).
10. B. MacIntyre, M. Gandy, S. Dow, and J. David Bolter: DART: A toolkit for Rapid Design Exploration of Augmented Reality Experiences, *Proc. of Conference on User Interface Software and Technology (UIST04)*, pp. 197–206 (2004).
11. H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana: Virtual Object Manipulation on a Table-Top AR Environment, *Proc. of the International Symposium on Augmented Reality (ISAR2000)*, pp. 111–119 (2000).