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
http://ubi.eedept.kobe-u.ac.jp/index_e.html

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 to utilize various sensors. In addition, since usually sensors and contents are deeply associated in contents, it is difficult to change sensors and to add sensors in such contents. In this paper, we propose a framework that helps content 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. Sensor management layer hides the difficulty in sensor management from content creators. Motion recognition tools and coordination support tool enable contents creator to use motion data in flash contents easily. We confirmed that creators can create interactive contents easier by using a prototype system.

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. Such intuitive interfaces have a potential to make complex computer operations easy. However, it is difficult for contents creators who create contents using *Flash* or other animation tools to develop such physical contents since they usually do not have the knowledge how to manage sensors and how 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 that is used in contents because the sensor management process is integrated to the contents. Therefore, there is a requirement on contents creation support system to manage sensors easily.

Here, Flash[6] that can provide interactive contents through web browsers is more and more popular all over the world. Flash content has various advantages such as small file size with respect to rich contents, and support for various platforms. Most people can play flash contents on his/her PC since *Flash Player* that runs flash contents is used by more than 99% people of mature market (US, UK, Canada, France, Germany, and Japan) in March 2009. Furthermore, flash is often used to create interactive contents since it is easy to program the key inputs function and to control images by using scripting language that is called *ActionScript*.

Thus, we propose a framework for flash creator to create interactive contents that are enjoyed by using sensors and actuators. Using our framework and implemented tools, the difficulty in sensor management is hidden from contents creators and they can create interactive contents easier. The remainder of this paper is organized as the follows. In Section 2, we describe our environmental assumptions. In Section 3, we present a brief summary of related works. Section 4 describes the process of implement contents with an example using our framework. We explain our framework in detail in Section 5 and discuss our framework in Section 6. Finally, we conclude our research 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 but can create flash contents. Recent days, there are many people that can create flash contents since flash becomes popular. Moreover, we employ *Wearable toolkit*, that we developed, to define/recognize user actions and contexts. The toolkit supports developing context-aware systems. It consists of an event-driven rule processing engine and some plug-ins that are implementation of functions. The rule in the toolkit is called ECA rule, and it consists of *Event*, *Condition*, and *Action*, called ECA rule. 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 and the position of station as *Condition*, and the navigation to web site of train schedule as *Action*, as shown in Figure 1.

Moreover, Wearable toolkit has *Context Definition Tool* plug-in with which users define the users motions and contexts easily, as shown Figure 2. After they demonstrate a motion that they want to define the system registers the motion. When they perform the same motion, the tool issues a motion recognition event. The Context Definition Tool supports 3-axis acceleration sensors, temperature sensors, GPSs, and RFID tag readers.

Figure 3 shows an example of our assumed content and user. We assume that users interact with the contents by their motions in front of the screen. They can receive feedbacks by the screen, sound, and some actuators.

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

Fig. 1. An example of ECA rule

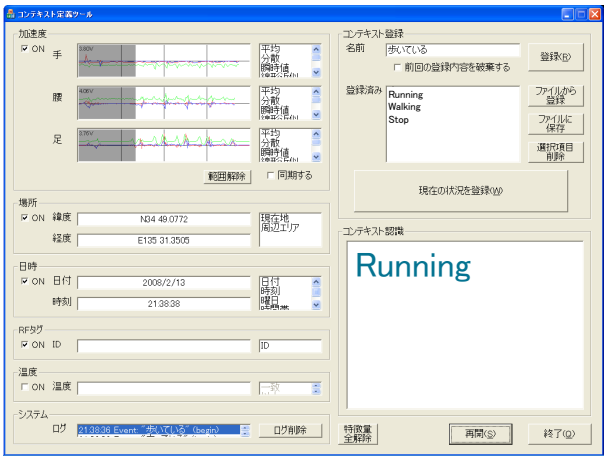


Fig. 2. A snapshot of context tool

3 Related works

There are many researches to develop services using various sensors and actuators easily.

Phigets[3] is a toolkit that consists of various sensors and various actuators such as buttons, sliders, mini-joysticks, touch sensors, RFID tag readers, motors, and LEDs. We can develop applications using the library that enable us to use Phigets in various development environments such as C++, Java, Max/MSP, and flash. Teleo[2] is also a toolkit made by MakingThings and we can develop applications that need sensors and actuators using it. Since it can output analog data by using Pulse Width Modulation, it can present fine degree. For example, it can control the rotating speed of motors. Gainer[8] is a hardware platform to connect various devices to computer easily. It cannot connect only general sensors and actuators but also new hardwares that we designed.

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 the computer and devices. Our framework supports the definition of user actions/contexts since we assume that the creators do not have enough knowledge to process the sensor data.

GlovePIE[1] is a tool that supports connecting existing device to existing applications. This tool assigns a function of the device to a function of the

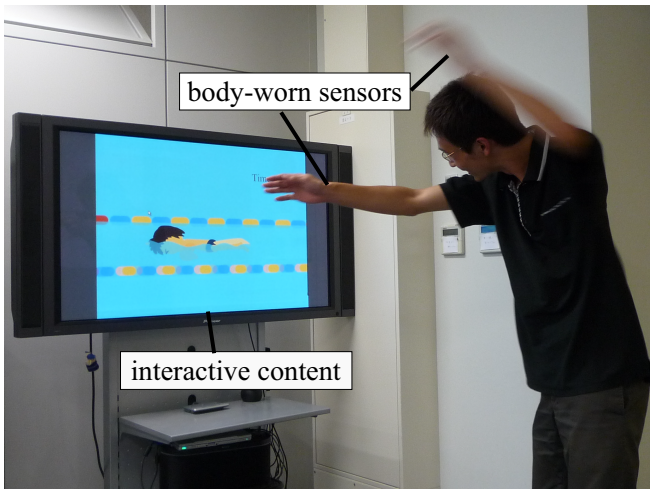


Fig. 3. An example of playing contents

application using a scripting language. Although we can develop the application that buttons and degrees of sensor data are assigned simply, it is difficult to develop the application that use actions recognized by sensor data.

A CAPpella system[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

First, we explain the overview of contents development using our framework. In this section, we show an actual content development of an interactive game content; a 110m hurdles game as shown in Figure 4.

Following steps show the development process of the contents.

Step 1 Design of contents
Step 2 registratoin of motions
Step 3 Description of rules for Wearable toolkit
Step 4 Creation of flash contents
Step 5 Debug and test

Especially our framework supports Step 2, Step 3, and Step 4 since we supposed that these steps are difficult for flash contents creator. Each step is detailed in follows.

Step 1 Design of contents

We established the overall picture of the contents in this step. Then, We discussed the purpose of the contents. Particularly on interactive contents, it is important to clarify what motion are used and which actuators are used.

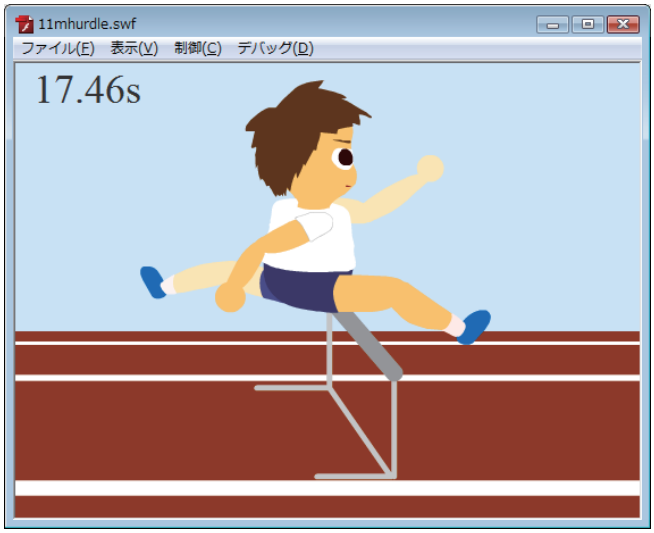


Fig. 4. A 110 meter hurdles game

In this case, the purpose of the game is to compete the time from start to goal. To be interactive contents, the character on the flash contents runs while the user runs, it stops when the user stops, and it jumps when the user jumps. Moreover, we design that a vibrator works when the character on the contents fails to jump a hurdle.

Step 2 Registration of motions

Next, we registered the user motions that are designed in previous step using Context Definition Tool. To register them, we only performed the actual motions and answer several questions presented by the tool. We also confirmed whether the action can be recognized accurately using this tool.

For 110m hurdles game, we registered the 3 motions; "run", "stop", and "jump" and confirmed whether they can be recognized accurately.

Step 3 Description of the rules

In this step, we described the rules for Wearable toolkit to communicate flash contents. Using our *Code Generation Tool*, whose detail is explained in following section, it is easy to describe the rules for Wearable toolkit. To create the rules, we only input the association between user actions and ActionScript that is activated when the motion is recognized. Concretely, to create the rules about "run" action, we input the name of action "running" and a script that is executed at running as shown in Figure 5. By doing this, the Code Generation Tool generates the scripts and the rules to execute the input scripts when the system recognizes "running". Then we also input descriptions about "stop" and "jump".

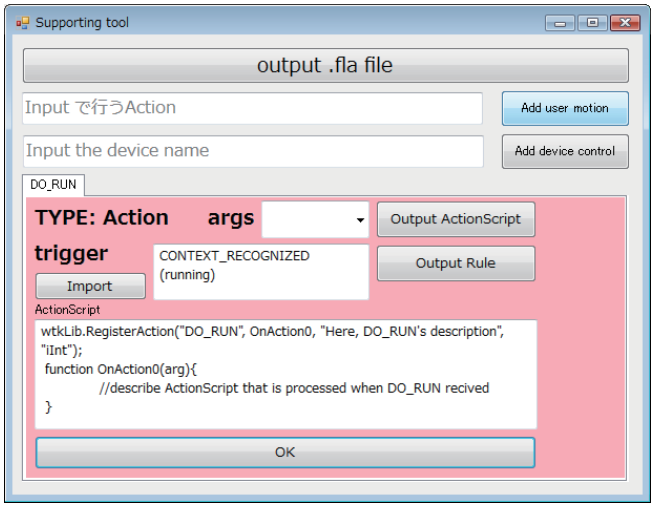


Fig. 5. Definition of the action which is executed in Running

In a same way, we also easily get the scripts and the rules to control actuators by using the Code Generation Tool. In concrete, we selected the vibration function from the list of Wearable toolkit functions, and named the function "VIBRATOR_VIBRATION". To control the vibrator from flash contents, we described this function call in the flash content. Figure 6 shows the definition of the device control.

Step 4 Creation of flash contents

In this step, we created flash animation contents. The Code Generation Tool generated flash project file in which needed script for communication is already described. We implemented the content based on this project file.

To control an actuator, we only describe RaiseEvent() function that has the defined event name as an argument at the line where we want to control it. In 110m hurdles game, since we wanted to vibrate when character hits hurdles, RaiseEvent() function was inserted into the code for collision detection.

Step 5 Debug and test

Finally, we debugged and tested the contents. We used the debugger of Wearable toolkit for the debug about communications. Using this debugger, we can confirm whether flash can receive the user motion event and Wearable toolkit can receive the device control event. Moreover, it is important for interactive contents to recognize user motions accurately. When we do not satisfy the accuracy of recognition, we can change the certainty factor or redefine the user actions.

We can change complete contents easily since the Code Generation Tool can generate rules and scripts individually for each actions and actuators.

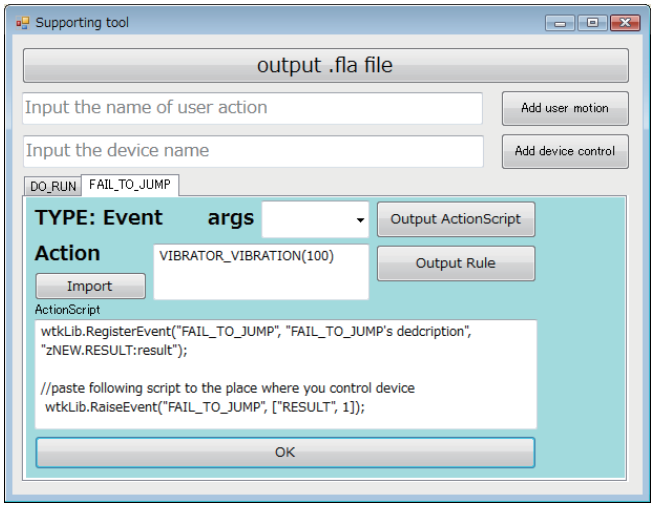


Fig. 6. Definition of the device control in the failure to jump

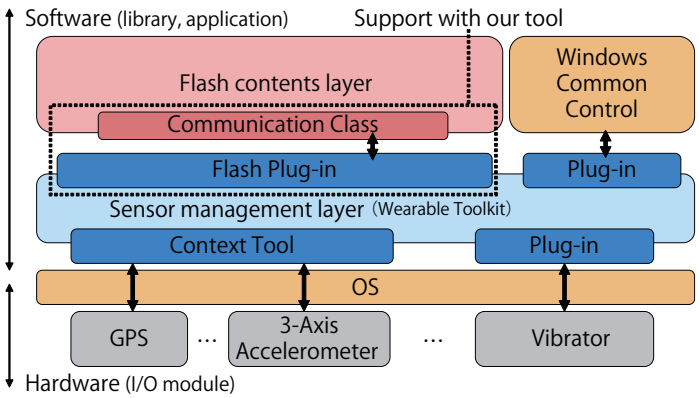


Fig. 7. Structure of proposed framework

5 Proposed framework

Our framework consists of 2 layers as shown in Figure 7. *Sensor management layer* that manages sensors and actuators, and recognizes user actions. *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. Using our framework, if they define the user motions that they want to use, flash contents can receive the motion-recognition event from sensor management layer. The sensor management layer enables us to define the user actions semi-automatically

using *Context Definition Tool* of Wearable toolkit at development stage. During execution, the tool recognizes the defined user motions from connected sensors.

In our research, we add a function for communicating with flash contents to Wearable toolkit and the implement a *Code Generation Tool* that generates the rules of Wearable toolkit and the scripts for flash contents for associating between motions and actions to support creating contents. In addition, we modify the Context Definition Tool to output the certainty factor of recognized motions.

The details of these implementations are described in the following subsections.

5.1 Communication between Wearable toolkit and Flash

To realize the communication between Wearable toolkit and flash contents, we employ the Flash Player's commands that is originally used for communication with Flash Player and the contents. In concrete, Wearable toolkit controls the Flash Player and communicates with the contents through the player. Though this command cannot transfer without text message. Text messages has enough capability to send context information to flash contents.

The other communication method that is used in various other related works such as Phidget and Gainer uses sockets communication. The advantage of this method is that the contents can receive stream data from the server, and it means that this method is suitable for processing the raw data from sensors. However, we do not suppose contents creators manage raw sensor data since this requires enough knowledge for sensor hardware and pattern recognition techniques. Moreover, to use this method, flash contents need complex scripts to establish and communicate using sockets. This is because we did not employ this method in our framework.

The communication between Wearable toolkit and flash occurs in 3 situations as shown in the followings:

Communication on flash contents starting First communication occurs in initializing contents. When a flash content starts, it sends messages to Wearable toolkit to register user motions and actuators that the content controls by using fscommand() function[4]. To send messages, the creators describes the scripts in initializing process as shown in Figure 8. The reason is that the creators want to define the motions and actuators on each contents since they are different each contents.

When the toolkit receives the messages, it registers the messages as events or actions of the toolkit. At the same time as sending message from the content, it associates the motion name with the scripts that is processed when the motion recognized. In the example as shown in Figure 8, ACTION_NAME action is associated with OnAction() function.

To get the initializing scripts, the creators can use Code Generation Tool. They input only the motion name, scripts that is executed when the motion recognized, the actuator that they want to control, and the trigger of controlling the actuator.

```
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. 8. Scripts that called at initialization of flash contents

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

Fig. 9. an ECA rule to send a message to flash contents

Communication when user motion is recognized After a flash content is started, Wearable toolkit works independently from flash contents. The toolkit constantly receives sensor data and tries to recognize user motions from sensor data. The flow of reorganization is shown as follows.

1. Sensor management layer recognizes user motions constantly.
2. When the registered motions are recognized, a ECA rules that generated Code Generation Tool is executed and send the recognition results to the flash content.
3. The flash content calls the associated scripts according to the received result.

In Step 2, the toolkit processes a ECA rule that is for communicate with flash contents. Figure 9 shows the ECA rules. The example shows when the context "CONTEXT_NAME" is recognized, the toolkit send a message "ACTION_NAME". Our Code Generation Tool generates the rule at the same time of generating initializing script.

To realize this communication, in *FLASH_ACTION* action, the toolkit calls SetVariable() function [5] on the SWFobject, that is flash content format. The function can change value of a variable on the flash content. In the content, the script are described as a callback function that is called when the variable is changed. In the example of Figure 8, OnAction() function is executed when the value of the variable is changed to ACTION_NAME.

Communication when content controls actuators Our framework enables flash contents to control actuators. When a flash contents controls the actua-

```
fscommand('EVENT_NAME', 'arguments');
```

Fig. 10. An example of fscommand

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

Fig. 11. ECA rules to control actuators

tors, fscommand() function on the contents is executed to send text message to Wearable toolkit. The flow of controlling actuators is shown as follows.

1. Flash content calls fscommand() function.
2. The ECA rule in Wearable toolkit, which is created by our Code Generation Tool, received command and controls associated actuators.

In Step 1, the fscommand() send the event name and the arguments of the event as shown in Figure 10. When the toolkit receives the event, in Step 2, the rule that named EVENT_NAME as shown in Figure 11 is processed. Our Code Generation Tool also generates this rule at the same time of generating initializing script.

6 CONSIDERATION

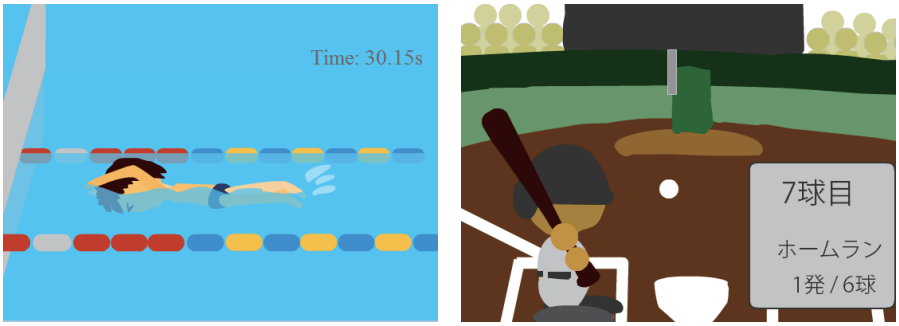
6.1 Evaluation of implementation time

We discuss the easiness of contents development by using our framework actually.

First we implemented an interactive game by modifying existing breakout game. We associated 2 motions on game with 2 physical motions. We took less than 1 hour to modify this game.

Next we implemented 2 games that are homerun chase game and swimming game. In homerun chase game, to swing the bat on game, a user swings an acceleration sensor. In swimming game, when user strokes with his arms, the game character swims forward. It took 2 hours for developing homerun chase game and 3 hours for developing swimming game. Most of the time for implementation was to create flash contents. Approximately 15 minutes are used for the definition of user motions and the description of rules that are needed to communications. Figure 12 shows snapshots for created games.

In addition, we created a donation box content for 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



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

Fig. 12. Created games

(a) waiting (b) bows

Fig. 13. A content for donation box

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 13.

This contents was also developed in approximately 3 hours. Thus, it needs little time to implement the contents using user motions.

6.2 Difference from existing flash contents

Through the implementation of interactive contents, we feel that the difficulty of games becomes high since the gesture inputs are difficult compared with the conventional input methods.

This problem happens on most interactive contents using physical motions. Therefore, we should adjust the game difficulty according to the recognition rate and the delay of the motions.

7 Conclusions

In this research, we implemented a framework that enables to communicate Wearable toolkit and flash contents and we confirmed that contents creator could develop intuitive contents by physical motions easily. Our framework enables not only to use motions for flash contents but also to control actuators from them. Furthermore, we implement Code Generation Tool to generate the scripts and the rules that are needed for communications.

As future works, we will let more flash content creators develop interactive contents to evaluate our framework.

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).