Dandelion Mirror: An Interactive Visual Design Using IoTtalk

Chung-Yun Hsiao[1(✉)], Chih-Chieh Huang[2], Yi-Bing Lin[1], and Yun-Wei Lin[1]

[1] Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan, R.O.C.
phoebe.cyhsiao@gmail.com, jyneda@gmail.com, liny@cs.nctu.edu.tw

[2] Center of General Education and College of Arts, National Tsing Hua University, Hsinchu 30013, Taiwan, R.O.C.
scottie.c.c.huang@gmail.com

Abstract. IoTtalk is a platform for IoT device interaction, which has been used to develop many IoT applications such as home automation. This paper uses Dandelion Mirror artwork as an example to show how interactive visual design can be conveniently implemented in IoTtalk. The Dandelion Mirror artwork conducts cyber physical interaction, which animates a dandelion in a mirror when a person smiles at that mirror. The dandelion grows larger as the person smiles bigger. The flower also vibrates following the heartbeat rate of the person. In our approach, the camera detecting the face expression and the heartbeat rate is considered as an input IoT device, and the mirror display for dandelion animation is considered as an output IoT device. In IoTtalk, the features of the dandelion animation are considered as the actuators that can be independently controlled. IoTtalk nicely connects these IoT devices, where various features can be created for interactive visual design with little or without any programming effort.

Keywords: IoTtalk · Interactive visual design · IoT platform
Cyber physical interaction

1 Introduction

Through cyclical and collaborative processes between people and technology, interactive design emphasizes a user-oriented field of study on meaningful communication of media [1–3]. An interactive design typically involves real-time interaction between human and machines. Several studies have focused on interaction between human behavior and visual objects, i.e., interactive visual design. Examples for such interactive visual design include Wii and Kinect game system, human-computer interaction, and so on.

This paper implements an interactive visual design called Dandelion Mirror. In this design, the smile expression and the heartbeat rate (HBR) of a person in front of the mirror affect the size and the shape of a dandelion animation displayed in the mirror.

Y.-B. Lin et al. (Eds.): IoTaaS 2017, LNICST 246, pp. 108–116, 2018.
https://doi.org/10.1007/978-3-030-00410-1_14

Dandelion Mirror conducts cyber physical integration, where smile and HBR are detected in the physical domain and the dandelion animation is illustrated in the cyber domain.

We use the Internet of Things (IoT) approach to implement Dandelion Mirror. In an IoT system, the IoT devices interact with each other through a network server following protocols such as AllJoyn, OM2M, OpenMTC [4] or any proprietary protocols. An example is a smart home appliance application where a remote controller (an input IoT device) controls home appliances such as a fan or a light (output IoT devices or actuators) through the IoT server/gateway. A program called network application is executed at the server to implement the desired interactions between the input and the output devices. This paper utilizes IoTtalk, an IoT platform [4, 5] that supports quick development of network application software. We use IoTtalk to implement Dandelion Mirror. Through this implementation, we will demonstrate that IoTtalk nicely connects the IoT devices, and through flexible connections, different features can be created for interactive visual design with little or without any programming effort. The paper is organized as follows. Section 2 introduces IoTtalk. Section 3 shows the implementation of the dandelion animation. Section 4 shows the implementation of the IoT device connections. Section 5 concludes our work with future research directions.

2 IoTtalk Concept and Usage

In IoTtalk, every IoT device is characterized by its functionalities or "device features". A device feature (DF) is a specific input or output "capability" of the IoT device. Consider a mobile phone as an IoT device. The input device features (IDFs) are sensors or controls that can be the microphone, the Gyro sensor, the GPS, the camera, and so on. The output device features (ODFs) are actuators such as the display screen, the speaker, and so on. An IoT device is connected to the IoTtalk server in the Internet through wireline or wireless technologies. The network application that defines the interaction between the IoT devices is developed and executed at the IoTtalk server. When the values of the IDFs are updated, an IoT device informs the network application to take some actions, and the network application sends the result to the ODF of the same or another IoT device to affect that output device. With this view, the IoT devices interact with each other through their device features.

The implementation of Dandelion Mirror in IoTtalk includes an input device called Intel-Cam and an output device called Dandelion. The hardware of Intel-Cam is an Intel RealSense Camera (model F200 or SR300), which can detect human facial expressions and the pulse rate. The detection functions are provided by Intel RealSense SDK [6]. In our approach, the connections of the input and the output devices can be easily established through the IoTtalk GUI. In this GUI, an input device is represented by an icon placed at the left of the window, which consists of smaller icons that represent IDFs, and an output device is represented by an icon placed at the right-hand side of the window, which includes ODF icons. For example, the icon in Fig. 1(a) represents the input device Intel-Cam. To implement the Dandelion Mirror application in IoTtalk, Intel-Cam detects the smile expression, and output the result through the *Smile* IDF (Fig. 1(c)). Intel-Cam also detects the heartbeats, and outputs the

measure through the *HBR* IDF (Fig. 1(b)). In Fig. 1(d), the Dandelion output device is a visual Java program installed in a Mirror display [7], which has two ODFs *Size* (Fig. 1(e)) and *Vibration* (Fig. 1(f)). By connecting the *HBR* IDF to the *Vibration* ODF (Fig. 1(1)), Dandelion will animate vibration based on the frequency of the measured heartbeat rate. By connecting the *Smile* IDF to the *Size* ODF (Fig. 1(2)), when the camera detects the smile expression, the flower grows bigger.

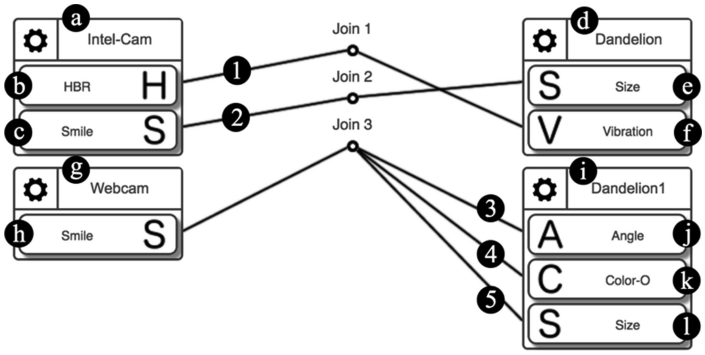

Fig. 1. The basic layout of Dandelion Mirror in IoTtalk GUI.

With IoTtalk, we can easily replace the input or the output devices of Dandelion Mirror with different implementations. For example, we can replace Intel-Cam by an ordinary webcam (Fig. 1(g)) connected to a computer or is built in a laptop that executes facial expressions provided by the OpenCV Haar feature-based cascade classifier [8]. The Webcam device can be considered as a simplified version of Intel-Cam with one IDF *Smile* (Fig. 1(h)). We also implement another output device Dandelion1 (Fig. 1(i)) with three ODFs *Angle* (Fig. 1(j)), *Color* (Fig. 1(k)) and *Size* (Fig. 1(l)), which can change the shape. The *Smile* IDF of Webcam is connected to the *Angle*, the *Color* and the *Size* ODFs to affect Dandelion1 (Fig. 1(3), (4) and (5)). Animations of Dandelion and Dandelion1 are implemented by the same Java program but with different parameter setups. Therefore, their appearances can be very different.

Through IoTtalk GUI, both Intel-Cam and Webcam are flexibly connected to either Dandelion or Dandelion1. Initially, the colors of both flowers are white. We may connect the *HBR* IDF of Intel-Cam to the *Vibration* ODF of Dandelion (Fig. 2(1)) and the *Angle* ODF of Dandelion1 (Fig. 2(2)). In this way, the heartbeats detected by Intel-Cam will control Dandelion's vibration and Dandelion1's angle. Both *Smile* IDFs of Intel-Cam and Webcam are linked to Join2 and then connected to the *Size* ODF of Dandelion (Fig. 2(3)). If the smile detected by Intel-Cam is bigger than that by Webcam, Join2 sends the value 1 (large flower) to the *Size* ODF of Dandelion. Otherwise the value 0 (no flower) is sent to Dandelion. Join3 links both *Smile* IDFs to the *Size* ODF of Dandelion1 (Fig. 2(4)). If the smile detected by Intel-Cam is bigger than that by Webcam, Join3 sends the value 0 to the *Size* ODF of Dandelion1. Otherwise the value 1 is sent to Dandelion1. With connections Join2 and Join3, big Dandelion appears and Dandelion1 disappears if the smile detected by Intel-Cam is

bigger than that by Webcam. On the other hand, if the smile detected by Webcam is bigger, then Dandelion1 appears and Dandelion disappears. Therefore, two persons can play a competition game with Dandelion Mirror. Join4 links both *Smile* IDFs to the *Color* ODF of Dandelion1 (Fig. 2(5)) so that if the smile detected by Webcam is bigger than that by Intel-Cam, the color of Dandelion1 turns yellow. Otherwise, Dandelion1 is white. To achieve the effects of Join2, Join3 and Join4, we need to program functions at these connections. We will elaborate on the details in Sect. 4.

We may also control the color of Dandelion1 by a color sensor as shown in Join5 (Fig. 2(6)). In this connection, the input device is a MorSensor [4] plugged in a color sensor.

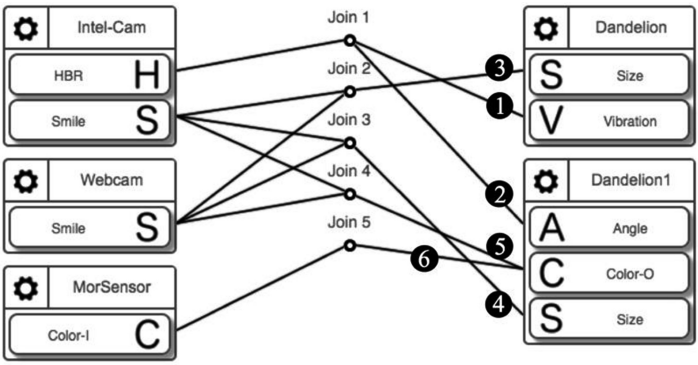

Fig. 2. An advanced layout of Dandelion in IoTtalk GUI.

3 The Dandelion Output Device

This section describes the output device Dandelion1 (Fig. 1(i)) with three ODFs *Size*, *Angle* and *Color*. In IoTtalk, the software for an output device consists of two parts. The actuator application implements the dandelion animation software. The device application (DA) provides the driver for wired or wireless communication to the IoTtalk server. The DA receives the data sent from the IoTtalk server and forwards them to the actuator application through the ODFs. These ODFs are specified in the Device Feature Window (Fig. 3). This window is provided by IoTtalk to manage the device features [4]. The Device Feature Window for the *Size* ODF (Fig. 3(1)) has one parameter (Fig. 3(2)) that represents the maximum level of the Dandelion to be grown. This parameter ranges from 0 to 10 (Fig. 3(3) and (4)). Figure 4 shows the Dandelion with *Size* = 8. The *Angle* ODF represents the growth direction of the Dandelion, which ranges from 0 to 120° (Fig. 3(5) and (6)). Figure 4 illustrates the dandelion with the *Angles* of 37° and 90°, respectively. The *Color* ODF represents the color of the Dandelion in the RGB format. That is, this ODF has three parameters, each of them ranges from 0 to 255 (Fig. 3(7), (8) and (9)).

The DA for the IoTtalk devices can be found in [5]. We use Java to implement the actuator application for the Dandelion animation that continuously draws graphical patterns to mimc the morphology of a dandelion. There are two types of parameters. The parameters for the display layout include the width (Fig. 4(1)), the height (Fig. 4 (2)), and the background color (Fig. 4(3)). As shown in Fig. 5, the parameters for the dandelion animation include the stalks s_0, s_1, s_2, s_3, the included angle between s_2 and the horizontal line (*Angle*) and the included angle between s_2 and s_3 (Angle1). These parameters are summarized in Table 1.

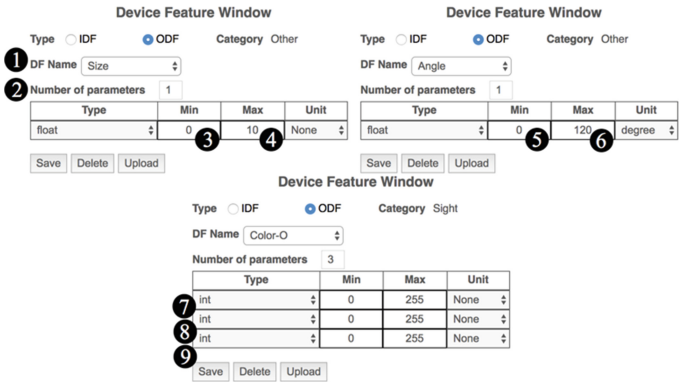

Fig. 3. Device feature window for Dandelion ODFs.

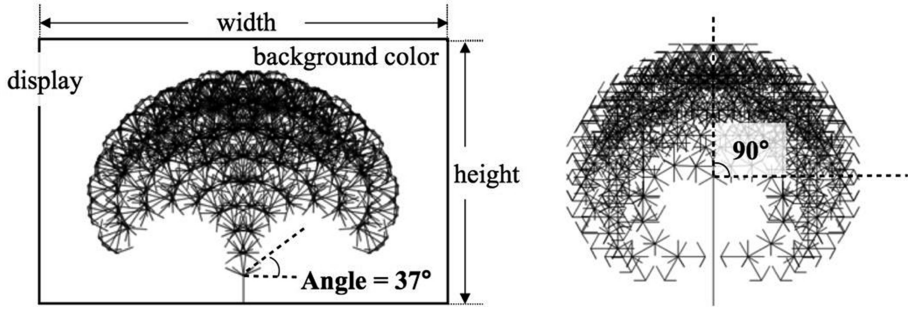

Fig. 4. Dandelion1 grows with different angle values (size = 8).

Dandelion is a fractal art implemented by a binary tree data structure in which each node has two children, the left child node and the right child node. Dandelion grows from the root node (level-0) denoted as $n_{R,0}$, which connects to stalk s_0 (see Fig. 5) with a default length of 384 pixels as defined in Table 1. For level $l \geq 1$, a level-l node is denoted as $n_{D,l}$ where $D = R$ if it is the right child node of its parent node, and $D = L$ if it is the left child node. Figure 5 shows the graphical patterns between the root node

$n_{R,0}$ and its children $n_{R,1}$ and $n_{L,1}$, respectively. The pattern consists of one stalk s_1, two stalks s_2 and two stalks s_3, which is a line with arrows at both ends. The length of stalk s_1 varies with level l expressed as

$$s_1(l) = s_1(0)(1 + 0.05l). \tag{1}$$

where $s_1(0)$ is the length of s_1 at level-0 with a default value 30 pixels as defined in Table 1. The other two stalks s_2 and s_3 have the default values of 17 pixels and 24 pixels, respectively. Angle1 has the default value 120°.

Table 1. The parameters of Dandelion.

	Parameter	Data type	Variable name	Default value
ODFs	Size	int	Size	0
	Angle	float	Angle	0.0
	Color	int	color_r	0
		int	color_g	0
		int	color_b	0
Display layout	Width	int	width	1024 pixels
	Height	int	height	768 pixels
	Background color	int	bg_color_r	255
		int	bg_color_g	255
		int	bg_color_b	255
Dandelion animation	Stalk s_0	int	s0	384 pixels
	Stalk s_1 at level-0	int	s1	30 pixels
	Stalk s_2	int	s2	17 pixels
	Stalk s_3	int	s3	24 pixels
	Angle1	float	a1	120°

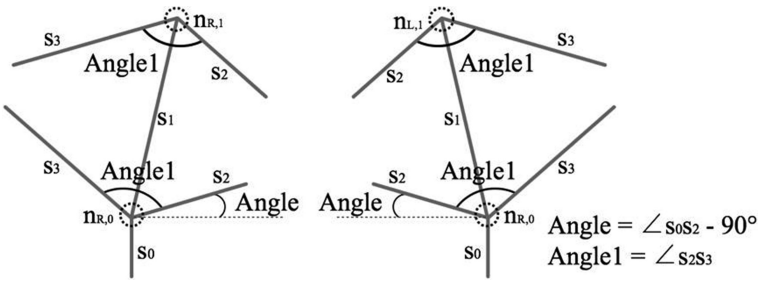

Fig. 5. Graphical patterns of Dandelion at level-0.

4 The Join Functions

IoTtalk allows one to select or implement functions at a Join point. The inputs of the function are the IDFs connected to the join point, and the output is sent to the ODFs connected to the join point. The user can select or implement very simple Join functions to create interesting I/O interactions. In Fig. 2, the function implemented at Join 2 has two inputs, i.e., the *Smile* IDFs of Intel-Cam and Webcam, and the result is sent to the *Size* ODF of Dandelion. By clicking the Join2 circle in the GUI, a window is popped up for the user to write a new function or select a predefined function [4]. The window consists of one or more IDF tables (Fig. 6(1)), one Join Function table (Fig. 6 (2)) and one or more ODF tables (Fig. 6(3)). For Join2, we select the predefined "larger than" function in the Join Function table (Fig. 6(4)), which is implemented in Python as

```
def run(a, b):
    if a > b: return 1
    else: return 0
```

where the input "a" is sent from Intel-Cam and the input "b" is sent from Webcam. Before the value 0 or 1 is sent to the *Size* ODF of Dandelion, it can be manipulated by another function in the ODF table. We select the "times10" function in the Size ODF table (Fig. 6), which is implemented as

```
def run(*args):
    return 10 * args[0]
```

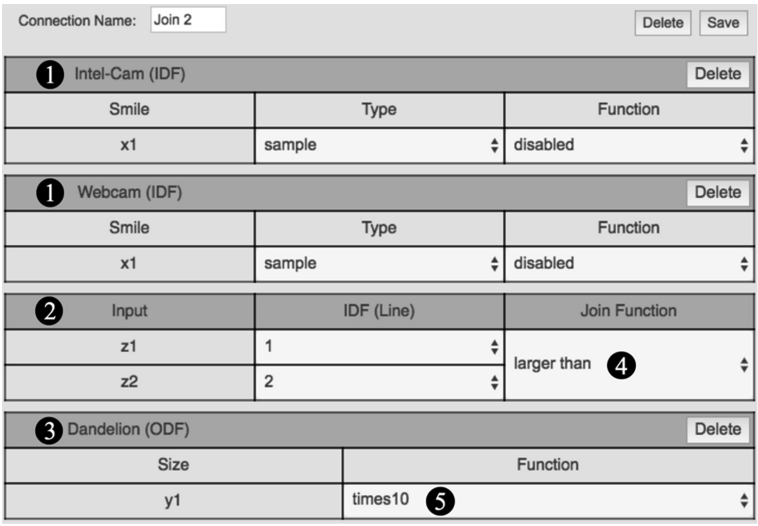

Fig. 6. The functions selected at Join2.

Therefore, if the smile detected by Intel-Cam is bigger than that by Webcam, the execution of these functions retunes the value 10. Otherwise, the value 0 is returned. The function implemented at Join3 is the same as that for Join2 except that the "larger than" function in the Join Function table is replaced by the "smaller than" function, and the final result reverses that of Join2.

5 Discussion and Conclusions

This paper designed and implemented the Dandelion Mirror artwork using IoTtalk. IoTtalk provides simple yet powerful connection mechanism for IoT device interaction, which can effectively and flexibly implement various features for cyber and physical interaction. There are two major components of this visual interactive artwork, a camera and a mirror that displays a dandelion animation. In IoTtalk, the camera is considered as a physical IoT input device, and the dandelion animation is a cyber IoT output device. We showed that with the concept of IDF and ODF, Dandelion Mirror can be conveniently implemented in IoTtalk. Through the ODFs, all features of the dandelion animation (angles, stalks, colors, size, vibration and so on) become the actuators that can be independently controlled in IoTtalk. Also, through IDFs, all detection mechanisms (for facial expression, color, heartbeat rate and so on) become the sensors that can be used to affect all actuators.

In the original design of this artwork, a dandelion in the mirror grows larger as a person in front of the mirror smiles bigger. With IoTtalk, we can easily modify Dandelion Mirror to create many novel and surprising ideas. For example, two sets of Dandelion Mirror can extend single person-dandelion interaction to multiple person-dandelion interactions. We showed how to modify the original Dandelion Mirror to a two-person competition game such that only the dandelion of the person with bigger smile will be displayed in his/her mirror. The IDF/ODF connections can also be easily modified to create multiple-person collaboration game on Dandelion Mirror.

Furthermore, many sensors and actuators have been connected to IoTtalk, and these IoT devices can be used to extend the interactive visual design. For example, the color sensor of MorSensor can affect the color of the dandelion animation. To conclude, we have demonstrated that IoTtalk nicely connects the IoT devices, and through flexible configurations, various features can be created for interactive visual design with little or without any programming effort.

Acknowledgements. The first version of Dandelion1 is implemented by Qing Liu.

References

1. Crawford, C.: The Art of Interactive Design. William Pollock, San Francisco (2003)
2. Kolko, J.: Thoughts on Interaction Design. Morgan Kaufmann, San Francisco (2011)
3. Saffer, D.: Designing for Interaction. New Riders, Berkeley (2010)
4. Lin, Y.B., et al.: EasyConnect: a management system for IoT devices and its applications for interactive design and art. IEEE Internet Things J. **2**(6), 551–561 (2015)

5. Lin, Y.B., Lin, Y.W., Huang, C.M., Chih, C.Y., Lin, P.: IoTtalk: a management platform for reconfigurable sensor devices. IEEE Internet of Things J. (2017). https://doi.org/10.1109/jiot.2017.2682100
6. Intel RealSense SDK Developer Reference. https://software.intel.com/intel-realsense-sdk/
7. Huang, S.C.: MSOrgm (motivational sensitive organism). Leonardo **42**(4), 374–375 (2009)
8. Open Source Computer Vision. http://opencv.org