

# ObjCube: Using Object-Oriented Metaphor to Review Interaction between Physical Objects

Rung-Huei Liang<sup>1</sup>

Wu-Chieh Tang<sup>2</sup>

<sup>1</sup> Dept. of Industrial and Commercial Design, National Taiwan University of Science and Technology, 43, Sec. 4, Keelung Rd., Taipei, 106, Taiwan, R.O.C. liang@mail.ntust.edu.tw

<sup>2</sup> Dept. of Computer Science and Information Engineering, Ming-Chuan University, 5 De Ming Rd., Gui Shan District, Taoyuan County 333, Taiwan, R.O.C. diprooo@gmail.com

**Abstract:** Designing metaphors in interaction design is to make users understand a system quickly based on their experiences. Object-oriented design principle is a programming style with objects, instead of data, as its metaphor. Originated from the physical world, object-oriented principle has developed into more complex form after these years of hard working by engineers in the digital space. Besides, on physical objects, more and more designers seek and apply fabulous metaphors to the attached digital information. However, concerning the foundation of interaction, in this research we tried to take the object-oriented principles as metaphor for designing interaction between physical objects. In addition to the physical properties of tangible things, digital properties are taken into account to adapt metaphor in virtual space. Presenting by physical computing, we introduce three basic concepts in object-oriented language, encapsulation, inheritance, and polymorphism, to the interaction of physical objects. To help us understand and think the possibility of redesigning the interaction of these digitally enabled physical objects, we demonstrate ObjCube as an example of this paradigm.

**Keywords:** physical computing, object-oriented, interaction design, metaphor, tangible media

## 1. Introduction

In the 2005 report published by ITU (International Telecommunications Union)[1], *The Internet of Things*[2], communications are classified into three categories, human to human (H2H), human to thing (H2T), and thing to thing (T2T). Similar to this classification, we think that it is also suitable to analyze types of interaction.

Interaction, in the past under the context of less supported technology and more social encounters, often referred to the human-to-human interaction. As computer getting popular, human-to-computer interaction (HCI) is emphasized, and therefore, “human-to-thing” interaction focuses mainly on HCI.

Although H2T interaction should not be limited by HCI, the digitally enabled objects indeed become more attractive than traditional ones. In general, the interaction between human and traditional objects is mostly restricted to the three dimensional space and physical properties. For example, interaction with a water glass is primarily through the mass, size, shape, and so on. On the other hand, digitally enabled objects broaden the scope of H2T interaction to a great extent, far beyond the physical interaction of tangible things. In a traditional situation when a water glass approaches a projector, the interaction between them is almost nothing. If the glass is attached with digital identification (e.g. RFID) and the projector is equipped with a reader, the interaction will immediately transfer from physical space to virtual space.

An important aspect of human-computer interaction is that under a specific context, humans interpret and manipulate machines. For instance, the flashing digits “12:00” on the screen of a videotape recorder might not represent any meaning. The recorder would not understand or decide what to do without human instructions. The machine is essentially passive. From the viewpoint of objects, when the digital abilities are quipped, objects can turn to communicate each other actively. Thus, to achieve the goal of smartly serving human needs becomes possible.

In the book “When Things Start to Think” [3], the smart coffee pot can recognize different cups and serve up preferences. Developed by researchers and students at MIT Media Lab, these coffee cups are attached with RFIDs and the coffee pot can communicate with them to identify preferences of owners. Therefore, there is no need for users to push any button during the whole process, and instead, the service is performed by thing-to-thing interaction rather than human-to-thing interaction. Our project aims at thing-to-thing interaction, in which we are especially interested in the interaction between “digitally enabled objects”. In general, the digital abilities are attached to physical objects through a set of metaphors. The projects “MusicBottles” [4] by Tangible Media Group at Media Lab is a good example to illustrate that the affordance of glass bottle can be abstracted as a metaphor for music playing. In 1981, the STAR system designed by Xerox, was thought to be the first graphic user interface. This system was originally designed for people in office environment, and therefore the icons on the display were all related to the office furniture. Hence, the metaphor abstracted from the workplace did help users to understand the interface.

The metaphor adapted by STAR system, is abstracted from the physical world and is applied in the digital space. Similarly, musicBottles also adapt metaphors from material world. These metaphors of both examples above are abstracted from the daily experience in the physical space. On the contrary, to explore metaphors for tangible object interaction, our research attempts to borrow metaphors in the virtual space, and then apply them to the physical space.

Object-oriented design principles are originated from physical space. Inspired by observing how people deal with objects in the material world, the structural programming languages are introduced with object-oriented principles to solve the growing complexity in code management. First appeared in 1967, this metaphor about how objects interact mutually has developed relatively complete and

well-known in digital space. Some of these principles are abstraction, encapsulation, polymorphism, and inheritance when designing codes. We think that the object-oriented metaphor is very suitable to apply on the interaction of physical objects.

Finally, we will illustrate the thing-to-thing interaction with a simple form. By physical computing [5], including microprocessor 8051, sensors, electronics, and tangible objects, we will demonstrate the interaction with object-oriented metaphor.

## 2. Related Works

In the physical space, as Kikin-Gil [6] proposed, the appearance and the method of manipulation of an object are affected by its physical properties. Bricks and ActiveDESK (Fig. 1) by Fitzmaurice et al. [7] allowed users to paint on a desk by dragging different shapes of bricks on the surface. Cylindrical bricks and cubic bricks would generate circles and squares on the desk, respectively. This system has successfully integrated the affordance of a tangible object and its function in a relative simple system of few operational instructions. However, as Kikin-Gil described, if the functions of a system become complex, this kind of physical interfaces will become a space-consuming machine without flexibility. To solve this problem, in our opinion, physical computing [5] is a good candidate to attach digital properties on physical objects while maintaining the physical interaction property of tangible objects.



Figure 1. Bricks and ActiveDESK[7]

In addition to physical computing, in order to search the paradigm of interaction design of physical objects, we take object-oriented design as a metaphor. This metaphor was first implemented as programming language by Ole-Johan Dahl and Kristen Nygaard to solve the simulation problem of how the different attributes from different ships could affect one another. Thereafter, this programming paradigm of simulating real world prevailed in the digital programming.

Sierra[8] presented three basic concepts in object-oriented programming: inheritance, encapsulation, and polymorphism. Preprocessed by abstraction of the problem, large project can be divided into several objects with these three basic concepts. After classification and designing of the attributes and behaviors within objects, the system can operate normally by mutual activation of objects themselves. However, this paradigm and style of program coding are not so intuitive to every programmer, especially the beginners. To help beginners to understand OOP, Feinberg[9] designed a tool to visualize code execution process

with animation of highly metaphorical illustration.

When OOP is taken as metaphor, interaction of tangible objects can be abstracted easily. In the projects by Things That Think [10] at Media Lab, Invisible Media [11], MusicBottles [4], Siftables [12], and The UbER-Badge [13] are all good examples to be interpreted with OOP metaphor. Moreover, these systems are all made for serving people by letting physical objects interact one another.

### 2.1 Digital Abilities of Physical Objects

To abstract the interaction of physical objects, we observe attributes and methods of objects and summarize them. As interaction considered, types of methods include presentation, communication, and sensing methods.

Presentation methods are necessary for people to understand the process and result of interaction. Typical devices to present the state of an object are LEDs, speakers, step motors, etc. Communication methods are the abilities to connect to internet, objects, database, etc. Wired or wireless connections are required to fulfill communication. To enable interaction with surrounding objects and environment, there must be sensing methods in the physical space. Electronic sensors are usually used to activate the proximity-triggered interaction.

### 2.2 Object-Oriented Programming Metaphor

Three basic concepts in OOP are concerned in this paper: encapsulation, inheritance, and polymorphism. After abstracting the simulation problem, encapsulation is to define an interface for others to access. For example, in Fig. 2, there are two attributes, rotational speed and gear ratio, and one method shiftgear to change these two attributes. In this example, we abstract transmission and encapsulate necessary attributes and methods in the object, and thus, an interface to access the encapsulated data is also provided. Analogously, when we design a physical object, we have to decide what attributes and method interfaces are necessary.

```
class transmission {
    int rpm; //rotational speed
    int gr; //gear ratio;
    void shiftgear(int rpm){
        rpm = rpm * 3.307;
        gr++;
    }
}
```

Figure 2. Class transmission

Inheritance is to create an object with properties of a defined object. In Fig. 4, class B inherits class A in Fig. 3. All the members in parent class A are inherited to class B. Moreover, class B can define other

attributes and methods additionally. As the material aspect considered, it would not be easy for physical objects to inherit from others. However, if the physical object is augmented by digital properties, the inheritance can be performed in digital domain.

```
class A{
    int i,j;

    void showij(){
        System.out.println("i and j: " + i + " " + j);
    }
}
```

Figure 3. Parent class A

```
Class B extends A{
    int k;
    void showk(){
        System.out.println("k: " + k);
    }
    void sum(){
        System.out.println("i+j+k: " + (i+j+k));
    }
}
```

Figure 4. Child class B

As well as inheritance, it is difficult to implement polymorphism for material objects. However, digital augmentation to a physical object can enable this concept. Polymorphism is understood as an interface with multiple forms of parameters, that is, the same name of method with different functions. A multi-functional steering wheel in Fig. 5 can be described as polymorphism in Fig. 6. Objects in material world are usually limited to fixed functions. For example, a projector is made for projection, CCD is for capturing images, and a mouse is for input. To fulfill this concept, we augment objects with digital properties.



Figure 5. Steering wheel

```
class control(){
    void steering(){
        turn_left();
        turn_right();
    }
    void steering(int seconds){
        honk(seconds);
    }
    void steering(int gr, int rpm, int UorD){
        gearshift(gr, rpm, UorD);
    }
}
```

Figure 6. polymorphism

### 3. Implementation

To illustrate our concept, we implement ObjCube. A paper box shown in Fig. 7 is a simple material

object. We attach a piece of LED array on the surface of this box. Inside the box is a simple I/O board, a physical computing platform, Arduino. Codes are uploaded to control the flashing LEDs. By implementing this computing technology, this material box is equipped with digital presentation method.



Figure 7. Simple material object



Figure 8. digitally enabled physical object

Regarding encapsulation, the object in Fig. 8 can be described as codes in Fig. 9. LED\_Number is attribute and Light( ) is a lighting method according to the form.

```
class ObjCube{
int LED_Number; //number of LEDs
Light(form){ }; // the form of lighting
}
```

Figure 9. Class ObjCube (encapsulation)

```
class ObjCube_OverloadDemo{
int LED_Number; // number of LED
int LED_Color; // LED color
int speaker_volume; //vol. of speaker
Light (form){ }
Ligh (form, color){ };
Sound(loud){ };
}
```

Figure 10. Polymorphism (overload)



Figur 11. Enhanced OjbCube

Fig. 11 shows an object that extends the object in Fig. 8. Since inheritance is hierarchical, the child class must be at least quipped with the same hardware devices as parent class does. In Fig. 12 and 13, the bottom box inherits all attributes and methods of the top box. Fig. 14 shows the child object with an additional LED array performs exactly the same LED flashing patterns from parent.



Figure 12. The beginning of inheritance



Figure 13. The end of inheritance



Figure 14. The interaction after inheritance

Finally, we demonstrate the application of polymorphism. Figure 15 and 16 show how we trigger the polymorphic process. The left box is overloaded from the right one. Through the polymorphism metaphor (as Fig. 10 shows), the left box in Figure 17 can act not only as the original ObjCube does but also those overloaded functions according to different types of activating parameters. In this case, different patterns of LED display are possible.



Figure 15. The beginning of polymorphism



Figure 16. The end of polymorphism



Figure 17. The interaction after polymorphism

#### 4. Conclusion and Future Work

This research is to extend the principle of TTT (Things That Think and Thing-To-Thing interaction), and make physical objects serve people in ambience. Introducing object-oriented programming language as a metaphor to our research, we attempt to interpret and redesign physical objects with digital abilities. Interaction of these objects can be easily observed and modeled.

Furthermore, we try to search a paradigm of designing digitally enabled physical objects. Through practice of this paradigm, especially for those with computer technology background, we

hope to reduce the difficulties and frictions of designing human-object interaction or physical objects in digital art, since we have successfully demonstrate these interactions with familiar OOP concepts.

Our work is to simplify the process of interaction design, based on a well-known and popular design principle, OOP. However, more intuitive interface for designers and users to plan and experience, such as management of private and public attributes, attachment and detachment of attributes, exception handling of unrecognized objects, and default or void feedback for invalid interaction, needs more study in detail in the future.

## References

- [1] International Telecommunications Union, <http://www.itu.int/net/home/index.aspx>
- [2] The Internet of Things, <http://www.itu.int/osg/spu/publications/internetofthings/>, 2005
- [3] Gershenfeld, N: 1999, *When Things Start to Think*, Henry Holt and Co.
- [4] Ishii, H., Mazalek, A., and Lee, J.: 2001, "Bottles as a Minimal Interface to Access Digital Information", *Conference on Human Factors in Computing Systems*, ACM.
- [5] O'Sullivan, D., Igoe, T.: 2004, *Physical Computing*, Course Technology PTR.
- [6] Kikin-Gil, R., 2004, "Metaphors for physical interfaces", [http://courses.interaction-ivrea.it/papers/papers/kikingill\\_ruth\\_MetaphorsForTangibleInterfaces.pdf](http://courses.interaction-ivrea.it/papers/papers/kikingill_ruth_MetaphorsForTangibleInterfaces.pdf)
- [7] Fitzmaurice, G. W., Ishii, H., Buxton, W.: 1995, "Bricks: Laying the Foundations for Graspable User Interfaces," *Conference on Human Factors in Computing Systems*, ACM.
- [8] Sierra, K., 2005, *SCJP Sun Certified Programmer for Java 5 Study Guide*, McGraw-Hill.
- [9] Feinberg, D.:2007, "A visual object-oriented programming environment", ACM.
- [10] Things That Think, <http://ttd.media.mit.edu/>
- [11] Maes, P., Merrill, D.:2005, "Invisible Media", MIT Media Lab.
- [12] Merrill, D., Kalanithi, J., Maes, P.:2007, "Siftables: Towards Sensor Network User Interfaces", MIT Media Lab.
- [13] Paradiso, J., Leibowitz, M.:2006, "Wearable Badge for Distributed Systems and Social Interactions", MIT Media Lab.