

Spatial Programming: Using the Physical World as a Computing System

Julien Pauty, Paul Couderc, and Michel Banâtre

INRIA

Campus Universitaire de Beaulieu

F-35042 RENNES

`julien.pauty, paul.couderc, michel.banatre @ irisa.fr`

Abstract. In this article we present spatial programming. We show how spatial programming can be used to create ubiquitous computing applications by tagging, with program code, physical processes that already exist. Spatial programming relies on the implicit organization of the physical space and the movements of the physical entities to ease the development of applications.

1 Physical process

Spatial programming [2] considers the physical world as a computing machine. The data used by this machine is the physical entities who populate the physical world, like: objects, people, buildings, rooms. . . This data is organized, like books on a shelf, articles in a shopping cart or containers in a boat. For example, consider a shopping cart. The content of the shopping cart represents its total price and inserting or withdrawing an article modifies its total price. In fact, inserting an article in the shopping cart is an addition; withdrawing an article from the shopping cart is a subtraction. Thus, the shopping cart is a physical calculator, yet limited to two operations. We consider that entities' movements correspond to computations.

The physical entities which populate the physical space represent the data. To be more precise, the information we mentally associate to the physical entities represents the data. For example, when I take a book from a shelf, I know that this is a book and not an anonymous object. I probably also know the author and maybe the content of the book. When I put this book on my desk, I also indirectly put on my desk the information I can mentally associate to this book [3]. Indeed, the entity associated with this information is now on my desk.

A physical process is represented by a flow of information in the physical world. This information is associated to the entities and moves with them. Our objective is to rely on this physical process to minimize the programming effort to create a ubiquitous computing application. We can consider that we map a digital process on a physical process, and that the control flow of the digital process is the control flow of the physical process. In fact, the physical processes are already there, we just augment them with digital computations.

2 Tagging physical computations with digital computations

The objective of spatial programming is to rely on the implicit computations done in the physical world to create simply ubiquitous computing applications. Spatial programming enables the programmer to create applications by tagging the physical computations with digital computations. When a physical computation happens, the corresponding digital computation is executed. Let us come back to the shopping cart example. We suppose that the shopping cart can detect the insertion and the withdrawal of the articles. The shopping cart tags the insertion of an article with `total += price` and the withdrawal of an article with `total -= price`. In this way, the shopping cart computes its total price automatically, according to the physical computations. Here, the application is just composed of code fragments that are executed when the corresponding physical computations happens. The flow of entities' movements is the application's control flow; there is *no* software control flow.

To execute the digital computations, we have to detect the associated physical computations. Since, entities are directly involved in physical computations, we choose to tag and detect the physical computations at the entities level.

2.1 Detecting physical computations

To detect physical computations each entity embeds a wireless device [5], which contains the data associated to the entity. An important point here is that the physical organization of the entities is analogue to the physical organization of the digital data. The physical world becomes a data store, where entities' movements imply data movements.

From the digital world, physical computations are seen as movements of digital data. Therefore, detecting physical computations is equivalent to detecting movements of digital data.

To detect a physical computation, an entity defines a volume. For example, the shopping cart defines a box which surrounds it. Physical computations imply that data items may leave or enter in this volume. For the shopping cart, prices enter or leave its bounding box. Thus, an entity tags the physical computations by associating digital computations to the entrance or the leaving of a data item from the volume. Our current implementation relies on a location technology, which enables an entity to detect the relative location of a data item.

2.2 Programming model

Spatial programming uses the physical world as a data store. An entity publishes a data item with the `out` operation. Spatial programming proposes several reading operations, which enable the entities to detect different kinds of physical computations: `read`, `take`, `readOnce`, `lostOne`. Detailing all the operations is beyond the scope of this paper, we just present `readOnce` and `lostOne` and illustrate them with the shopping cart application.

The `readOnce` and the `lostOne` operations define an addressing volume, relatively to the location of the entity which executes the operation. The `readOnce` operation stays blocked until a *new* data item enters in the addressing volume and returns this new data item. It enables the entities to detect physical computations such as: “a *new* entity has entered into the addressing volume”. The `lostOne` operation stays blocked until a data item has left the addressing volume and returns this data item. It enables the entities to detect physical computations such as: “an entity has left the addressing volume”. The lost data item must have been read previously with `readOnce`. The program code that is executed after an operation is released represents the digital computation associated to the detected physical computation.

For the shopping cart application, each article initially publishes its price with an out operation. To monitor physical computations, the shopping cart executes two threads, one for articles insertions and one for articles withdrawals:

```
while(1) {                               |   while(1) {
    price = readOnce(bounding_box);       |       price = lostOne(bounding_box);
    total += price;                       |       total -= price;
}                                         |   }
```

Spatial programming has been used to develop several applications, like Ubibus [1], which is an application to help blind people to take the bus.

3 Discussion

Spatial programming maps applications on physical processes. This mapping is performed by tagging physical computations with program code. The application execution flow is directly controlled by the entities’ movements. Therefore, to develop a spatial application we just have to develop the program code associated to each physical computation. Moreover, applications are mapped at the entity level, so spacial programming avoids the costs of creating a global digital model and maintaining the consistency between the model and the physical world. A side benefit of tagging a physical process with program code is that the physical process is not modified. Therefore, the user interacts with the physical process in the same way and implicitly interacts with the digital process [4].

References

1. M. Banâtre, P. Couderc, J. Pauty, and M. Becus. Ubibus: Ubiquitous Computing to Help Blind People in Public Transport. In *Mobile HCI 2004*, pages 310–314, 2004.
2. P. Couderc and M. Banâtre. Ambient Computing Applications: An Experience with the SPREAD Approach. In *Annual Hawaii International Conference on System Sciences (HICSS’03)*, 2003.
3. H. Ishii and B. Ullmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *CHI*, pages 234–241, 1997.
4. A. Schmidt. Implicit Human Computer Interaction Through Context. In *Personal Technologies 00*, 2000.
5. M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.