PetriNect: A tool for executable modeling of gestural interaction

Romuald Deshayes, Tom Mens, Philippe Palanque

PetriNect: A Tool for Executable Modeling of Gestural Interaction

Romuald Deshayes, Tom Mens
Service de Génie Logiciel, Université de Mons
7000 Mons, Belgique
Email: romuald.deshayes,tom.mens@umons.ac.be

Philippe Palanque
ICS-IRIT, Université Paul Sabatier
31062 Toulouse, France
Email: palanque@irit.fr

***Abstract*—In this showpiece we demonstrate PetriNect, an instance of a generic layered framework that we have developed for the specification and use of executable models of gestural interaction with virtual objects. The framework is built on top of Petshop and uses ICO models, a variant of high-level Petri nets. PetriNect uses the Kinect as input device for allowing the user to interact gesturally with virtual objects. We present two simple proof-of-concept prototype applications that have been developed for the purpose of this showpiece: a simple Pong game, and the interaction with a virtual bookshelf.**

I. Introduction

Mouse and keyboard have been the main devices for human-computer interaction for a few decades. A lot of effort has since been dedicated to come up with new means of interacting with computers. A recent trend consists of controlling the computer using natural interaction such as unconstrained human gestures. While many new applications using gestural interaction arise, few of them propose a unified way to internally represent the gestural interaction. Integrating such interaction techniques within a software application is often a real burden for developers, because of the technical details involved in the implementation and because of the accidental complexity of the tools and languages used. Visual modeling languages can hide an important part of the technical aspects involved in the development process, thus allowing a faster and less error-prone development. Executable modeling even reduces the need of compiling and maintaining lower-level source code altogether.

In a short article published at VL/HCC we have presented a generic and reusable layered framework for the specification of gestural interaction with virtual objects [1]. This framework has been realized using Petshop, a tool dedicated to the specification and execution of interactive systems based on ICO, a powerful model-based approach relying on high-level Petri nets [2]. The main aim of this framework is to facilitate prototyping of applications requiring gestural interaction, through the use of visual, executable models.

In this showpiece, we focus on PetriNect, an instantiation of the aforementioned framework customized to using the Kinect (`www.xbox.com/kinect`) as gestural input device to recognize and track hand movements of the user. This showpiece illustrates by means of a video and a live demonstration how to use PetriNect to model the interaction with virtual objects and execute these models. A preliminary video can be found here [3].

The PetriNect showpiece is illustrated by means of two simple prototype applications that serve as proof-of-concept:

- A Pong game (figure 1), in which the player needs to prevent the opponent (the computer or another human player) from returning the ball with his paddle. To interact with the game, the player simply uses single-hand gestures instead of a joystick.
- A virtual book shelf (figure 2), from which the user can drag books, open them, and turn pages, by using hand gestures, synchronized with the position of the head.

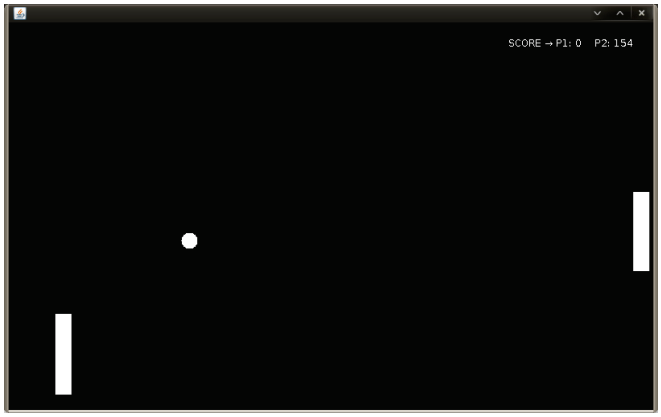

Fig. 1. A Pong game

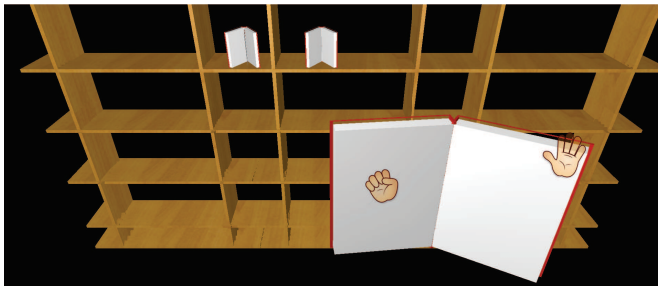

Fig. 2. A virtual book shelf

II. The PetriNect tool

The PetriNect instantiation of our framework receives gestural input from the Kinect 3D sensor that provides information about movement of body limbs.

A client-server approach is used to achieve modular communication between the Kinect and the framework. Raw input data regarding position of hands and head is captured by the Kinect, at a framerate of 30Hz, in a client application developed in C++ and sent to a Java server following a dedicated protocol. The server parses the data and encapsulates it in Java objects. An observer design pattern is used to transfer the Java objects to the first layer of the framework, inside Petshop (at model level). In this first layer, the positional information about hands and head is merged. The merging performs a transformation to match the head's coordinate system, so that the position of the hands is expressed with respect to the head's position.

The information regarding absolute 3D position and status (open or closed) of the users' hands is converted [4] into abstract events in a stepwise fashion, by the different executable ICO models of the framework. Layer 2 transforms the absolute position into relative movements by analyzing two consecutive data frames. This layer detects the state changes of the hands and interprets low-level gestures based on these relative movements. Layer 3 then interprets higher-level gestures by combining the low-level ones with information about the user's state.

The top layer of the framework, aimed at combining all provided gestures into models of gestural interaction with virtual objects, sends the necessary output to the target application (such as a graphical user interface or 3D rendering engine). As each virtual object is specified by an ICO model, a Java client is implemented at the code level to convert these models into output events that are sent, by means of an observer design pattern, to a Java server linked to the external target application.

Regarding our two prototypes, a Java server has been implemented for the Pong game, and a C++ server for the virtual bookshelf. For both applications, the data received from the framework are textual commands with some parameters (e.g. amount of movement). The Pong game has been developed using the Java Swing API. When a *move* event is received, the user's paddle moves according to the position of the hand. In the virtual bookshelf prototype using the Ogre3D graphical engine, different commands are received and can trigger visual feedback such as grabbing a book, opening it or turning the pages. The setup of the 3D scene and the animations are created with Ogre 3D, but all the logic depends on ICO models. For example, while opening the book, it is the role of the model to set the maximum opening angle of the book.

III. Provided gestures

The top layer of the framework contains a different model for each type of object in the virtual scene the user wants to interact with. These objects can be controlled using the gestures provided by layer 3. The same gesture can be interpreted differently by different objects, e.g., picking up a book from the shelf or opening a door can be performed by closing one hand and moving it towards one's chest.

The high-level gestures that we have currently provided to specify the interactive behaviour of virtual objects are:

- **Open** and **Close** are triggered when the user opens or closes a hand; a parameter specifies which hand.
- **Move** an open hand; a parameter specifies which hand and its 3D direction.
- **Drag**: move a closed hand; a parameter specifies which hand and its direction.
- **ExpandClose**: move both hands away from each other; a parameter indicates the direction of both hands.
- **ShrinkClose**: move both hands towards each other.
- **ColinearDrag**: move both hands in different directions while keeping them colinear.
- **NonColinearDrag**: move both hands in different directions but not colinearly; a parameter indicates the direction of both hands.

All these gestures can be analyzed in a short time interval (essentially, two consecutive updates received from the input sensor). These gestures have no concrete behavior associated to them, and can therefore be applied in different ways to different types of virtual objects.

The list of provided gestures is inevitably incomplete, as it is impossible to enumerate all possible gestural interactions using two hands. Our framework facilitates, however, the specification of new high-level gestures as a combination of more primitive ones. For example, we could combine the moving of both open hands to create a new gesture similar to ExpandClose. In such a way, one can create a wide range of behavioral models for an unbounded number of different types of objects.

IV. Conclusion

PetriNect is a fully functional prototype for executable modeling of interactive applications, relieving the developer from writing complex and statically compiled code. PetriNect is an instantiation of a generic, layered and modular framework for the visual specification and execution of gestural interaction models. High-level Petri nets, incarnated as ICO models in Petshop, proved to be particularly suited for expressing such models. Our proof-of-concept prototypes using the Kinect gestural input device show the practical feasibility of using our framework for gestural interaction.

Acknowledgement

The first author of this showpiece is financed by a FRIA scholarship.

References

[1] R. Deshayes, T. Mens, and P. Palanque, "A generic framework for executable gestural interaction models," in *Proc. VL/HCC*, 2013.

[2] D. Navarre, P. Palanque, J.-F. Ladry, and E. Barboni, "ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability," *ACM Trans. Comput.-Hum. Interact.*, vol. 16, no. 4, pp. 18:1–18:56, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1614390.1614393

[3] R. Deshayes, "YouTube video," YouTube, 2013. [Online]. Available: http://youtu.be/m9NIvZpQyjs

[4] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with Kinect sensor," in *ACM Int'l Conf. Multimedia*, 2011, pp. 759–760.