# Exploring Modeling Language for Multi-Touch Systems using PetriNet

**Francisco R. Ortega**
Florida International University
Miami,FL. USA
forte007@fiu.edu

**Malek Adjouadi**
Florida International University
Miami,FL. USA
adjouadi@fiu.edu

**Frank Hernandez**
Florida International University
Miami,FL. USA
fhern006@fiu.edu

**Su Liu**
Florida International University
Miami,FL. USA
sliu002@fiu.edu

**Armando Barreto**
Florida International University
Miami,FL. USA
barretoa@fiu.edu

**Naphtali Rishe**
Florida International University
Miami,FL. USA
ndr@acm.org



## Abstract
We are motivated to find a flexible and compact modeling language for multi-touch gesture recognition.We explored a modeling language for multi-touch gesture recognition using Petri Nets. We demonstrated that a Petri Net can be used for gesture detection. We explore the possibility of a model composed of many Petri Nets.

## Author Keywords
Multi-Touch, Gesture Recognition, Petri Nets

## ACM Classification Keywords
H.5.2 [Input devices and strategies]: Interaction styles

## Introduction
We are inspired by a couple of recognition systems for multi-touch devices. For example: proton++[6] and Gesture Coder[9]. We continue to be inspired by the simplicity of the $1 algorithm by Woobrock et al.[15]. We are motivated to continue to look for a more extensible, simple and uniform gesture recognition method that allows developers and HCI researchers to easily integrate it into their systems. In addition, our approach will be used for studies with different input devices.

In our initial exploratory approach towards a new modeling language for input devices (e.g., Multi-Touch), we present

**IRML** (Input Recognition Modeling Language). While this approach currently uses multi-touch devices only, the system will be extended for additional input devices.

We based our contribution in the state of the art for multi-touch gesture recognition. In addition, our work uses well established computer science theory, such as finite-state machines, regular expressions (RegEx) and Petri Net (PN)[14, 4]. For IRML, we use Petri Nets to represent multi-touch gestures.

## Background

Formalism for input systems can be found in Newman's pioneer work (1968), using a state diagram to represent a graphical system[12]. In 1990, the seminal work by Buxton in *a Three-State Model of Graphical Input*[1] used finite-state machines (FSM). The same year, Myers published a well-rounded model for input[10].

Formalism for input multi-touch gesture detection or touch events has also been explored. Recently, proton and proton++ showed the use of RegEx to accomplish gesture detection[6]. The use of state-diagram for multi-touch events was shown by Lao et al[7]. Kammer et al. used context-free grammar (CFG) to describe multi-touch gestures at a high level[5]. Gesture coder[9] creates FSMs by demonstration to later use them to detect gestures. Gesture Works and Open Exhibits by Ideum (Open Exhibits has a free non-open source product and Gesture Works sells commercial libraries) have a high-level language description using XML, called GestureML (or GML)[1]. Midas' framework[13], utilizes a rule-based language to define gestures.

Petri Nets have also been used to detect gestures. Nam et al. showed how to use colored Petri Nets to achieve hand (data glove) gesture modeling and recognition[11], using Hidden Markov Models (HMM) to recognize movements before feeding them to the PN. Our approach is created for multi-touch devices (at this stage) with the ability to define gestures without HMM. Our vision is a general modeling tool for input devices. Petri Nets exist in many flavors and they are widely used in the industrial and communication fields[16].

## Exploring IRML

We define a few terms and concepts about multi-touch and Petri Nets. A **touch point** (*tp*) is a representation of a 2D point on the screen. A **trace** ($T$) is a set of continuos touch points. Each trace has an ID (e.g., $tp_1$) and a state which can be down, moving, or up. For each trace there can be one down state, many move states, and one up state. A Petri Net is visually represented with tokens (small solid gray circle), transitions (rectangle), places (large circle), and arcs connecting places to transitions.

Originally, we wanted to expand the work by Kin et al.[6] using RegEx. However, we found a couple of reasons to use Petri Nets instead. First, some gestures can be difficult to implement with RegEx. Take a gesture where the angle produced by two touches must be maintained in a given range while performing the action. For example, given a set of points forming lines, the angle between those must be between 30 to 40 degrees. The reason why this is complicated to represent in RegEx is that the attributes are defined for each trace (touch). A possible solution may be to group the traces, but this makes RegEx become lengthy. Second, RegEx while useful, can become large for complicated models. Another approach that we looked into was FSM. FSM can become larger

[1] www.GestureML.org

than PN to accomplish the same tasks. One can represent a FSM with a PN but not the other way around. Therefore, Petri Nets were chosen.

Figure 1 shows a pseudo Petri Net that will work with high level Petri Nets, such as Coloured Petri Nets (CPN)[4], first-order-logic PN[8] and Priority Petri Nets (see [2] for types of PN). This figure is an example of a Petri Net for the two-finger swipe gesture. The swipe gesture consists of the user moving the fingers in the same direction. In the example, we use each trace as the token. This behaves as a trace pointer at the position of the current sample. It is possible to design our proposed PetriNet using the elements of the set T, similar to [6]. In other words, the tokens will be individual trace events such as 'dmmmu' for down, move three times and up one time. However, this makes the design lengthier and more prone to error. Figure 1 has additional labels (*A* to *E*) to simplify the discussion.

The two-finger swipe gesture requires two tokens to move forward. *Transition A* determines if the tokens are in down state. If they are, they move to the *PROCESS Place*. Once in process, the Petri Net determines the next path by using one of the correct transitions. For example, if the state of a given trace is 'up', then it will move to the *EXIT Place* via *Transition D*. If the two fingers are not moving in the same direction, then the tokens are consumed via *Transition E* to the *EXIT Place*. Otherwise, it will move to the *SWIPE Place*, via *Transition B*, to trigger an action for this event. The token will automatically move forward, using *Transition C* for the next iteration. It is important to notice that this example represents a pseudo Petri Net allowing for conflicts to be resolved[2].

From this basic example, we can construct a series of Petri Nets. Given a set of PN, named P, we can construct a model P = $(PN_1, PN_2, PN_3, ...PN_n)$. Once the model is constructed, it can be executed. Once the gesture is complete, the model can be reset. For the implementation, the model is stored in an XML file to be loaded and executed in the language of choice (e.g., C++, Java). However, at this time, we have not tested multiple gestures simultaneously.

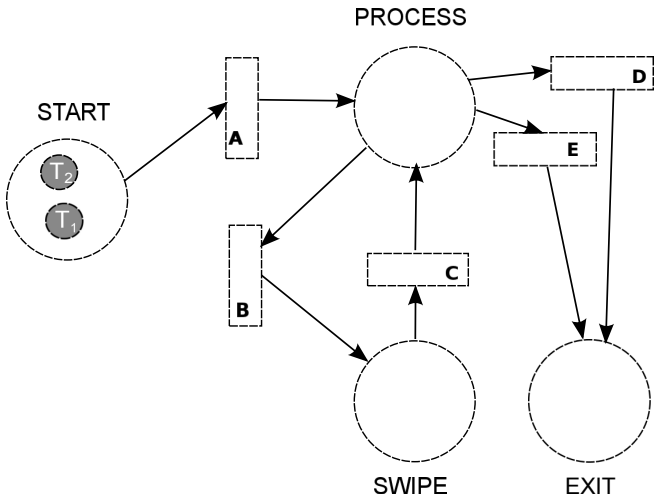


**Figure 1:** Two-Finger Swipe Gesture Petri Net.

## Conclusion and Future Work

We have shown an alternative solution to multi-touch gesture detection using Petri Nets. We have shown that the transition and high-level Petri Nets (e.g., CPN) can provide a powerful, flexible and compact approach to solve input detection in general.

Our future work will consist of two tasks. Creating a full-fledge Petri Net framework that will work with different input devices, such as multi-touch, gyroscopes,

and Leap Motion. This will include the pseudo-code and concrete code (e.g., C++) required to run a full-fledge system. In addition to IRML, our second tasks involves a visual approach using domain-specific language (DSL) as demonstrated in other domains, such as game development[3]. The DSL approach will include usability testing with developers. Our work is made with the objective to test 3D Navigation.

## Acknowledgements

This work was sponsored by NSF grants HRD-0833093, CNS-0959985, CNS-0821345, and CNS-1126619. Frank Hernandez and Francisco Ortega are recipients of GAANN fellowships (US Department of Education). Mr. Ortega is also a recipient of the McKnight Dissertation Fellowship.

## References

[1] Buxton, W. A three-state model of graphical input. *Human-computer interaction-INTERACT 90* (1990), 449–456.

[2] David, R., and Alla, H. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, Nov. 2010.

[3] Hernandez, F. E., and Ortega, F. R. Eberos GML2D: a graphical domain-specific language for modeling 2D video games. In *DSM '10: Proceedings of the 10th Workshop on Domain-Specific Modeling*, ACM (Oct. 2010).

[4] Jensen, K., and Kristensen, L. *Coloured Petri Nets*. Basic Concepts, Analysis Methods and Practical Use. Springer, 1996.

[5] Kammer, D., Wojdziak, J., Keck, M., Groh, R., and Taranko, S. Towards a formalization of multi-touch gestures. In *ITS '10: International Conference on Interactive Tabletops and Surfaces*, ACM (Nov. 2010).

[6] Kin, K., Hartmann, B., DeRose, T., and Agrawala, M. Proton++: a customizable declarative multitouch framework. In *UIST '12: Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (Oct. 2012).

[7] Lao, S., Heng, X., Zhang, G., Ling, Y., and Wang, P. A gestural interaction design model for multi-touch displays. 440–446.

[8] Liu, S., Zeng, R., and He, X. PIPE-A Modeling Tool for High Level Petri Nets.

[9] Lü, H., and Li, Y. Gesture coder: a tool for programming multi-touch gestures by demonstration. 2875–2884.

[10] Myers, B. A. A new model for handling input. *ACM Transactions on Information Systems (TOIS) 8*, 3 (1990), 289–320.

[11] Nam, Y., Wohn, N., and Lee-Kwang, H. Modeling and recognition of hand gesture using colored Petri nets. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 29*, 5 (1999), 514–521.

[12] Newman, W. M. A system for interactive graphical programming. 47–54.

[13] Scholliers, C., Hoste, L., Signer, B., and De Meuter, W. Midas: a declarative multi-touch interaction framework. 49–56.

[14] Sipser, M. *Introduction to the Theory of Computation*. Cengage Learning, June 2012.

[15] Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: a $ 1 recognizer for user interface prototypes. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM (Oct. 2007).

[16] Zurawski, R., and Zhou, M. Petri nets and industrial applications: A tutorial. *Industrial Electronics, IEEE Transactions on 41*, 6 (1994), 567–583.