International Journal of Science and Engineering Applications
Volume 2 Issue 4, 2013, ISSN – 2319-7560 (online)

Multi-Touch Gesture Application Using QT for Linux/Windows Operating System

Arnav Sonara
Computer Department
MIT AOE, Alandi(d)
Pune, India

Ajinath Funde
Computer Department
MIT AOE, Alandi(d)
Pune, India

Sumit Jadhav
Computer Department
MIT AOE, Alandi(d)
Pune, India

Swapneel Lodha
Computer Department
MIT AOE, Alandi(d)
Pune, India

Om Shankar
Computer Department
MIT AOE, Alandi (d)
Pune, India

Abstract: Now a days, user interaction considers time as its primary factor. Using menu bar, tool bar for accessing the actions to be performed take much time. So, the main purpose of this paper is to demonstrate how to minimize the access time of various features of the application by multi touch. It makes user interaction much easier by accessing various features using multi touch gestures. We are introducing idea for gesture support on applications like Image Viewer, Document Viewer, etc. We are making use of *Pan, Pinch, Swipe, and Rotate…etc.* gestures to make user interaction easier and faster. This idea can be implemented using C++ Qt Framework, developed by Nokia as it has great support for GUI and Gesture programming. The main advantage of Qt is that it is cross platform, once coded, it can be deployed on various operating systems.

Keywords: Input devices and strategies, Touchpad, Multi-touch, Gestures, widgets, Computer Access

1. INTRODUCTION

1.1 Motivation

THIS PAPER IS MOTIVATED BY TWO PRIMARY CONCERNS, THE NEED FOR BETTER DESKTOP COMPUTER ACCESS, PARTICULARLY FOR LINUX/WINDOWS OPERATING SYSTEM AND SECOND IS PLATFORM INDEPENDENT. TO REDUCE THE EXCESSIVE USE OF MOUSE & CURSOR METHOD OR KEYBOARD IS THE MAIN GOAL OF THIS PROJECT.

1.2 Need of Multi-touch

Accessing to computer and information technology is increasing dramatically in present scenario. So, time is the primary factor to access these resources. Accessing these features using mouse and cursor method takes much time. As in present scenario, there is great revolution in touchpad, they are able to detect up to ten fingers and we can make use of this feature to implement various gestures which can be then further used in various applications to trigger some action to be performed.

1.3 Why C++ Qt Framework?

Qt has great support for GUI and Gesture programming. Also it is cross platform application framework and makes use of standard C++. It has inbuilt tools like *"Meta Object Compiler"*and *"qmake"*. "*Meta Object Compiler"* is used for automatic code generation and "*qmake"* is used for compiling project on different platform without changing the code. It has vast number of predefined classes which can be used for various purposes during development of applications. The "*QObject*" is the root class of all other classes of Qt and which is mandatory for "*SIGNAL-SLOT*" functionality to work properly. It also contain UIC compiler which is used to convert ".ui" file generated by Qt Designer into C++ header file. It contains *Qt Assistant* which provides documentation for various predefined classes and functions already defined in Qt. So while implementing idea presented in this paper we can easily make use of this documentation and thus more user friendly than any other framework.

1.4 How Qt Supports Gesture Programming?

For events, Qt has library called *"QEvent"*. There is one function *"eventFilter()"* function in "*QAbstractEventDispatcher"*class that is used to detect whether the particular input is an event or not. For gesture events *enum* is defined in *"QEvent"* class as *"QEvent::Gesture"* and various other events like keyboard event, mouse event, etc. If no event is recognized then index returned by *"eventFilter()"*function is NONE if any gesture is recognized it returns index 198. After recognizing gesture events it uses "*QGestureRecognizer*"class which inherits *"QEvent"* class and *"QGesture"* class.For implementing various gesture it has predefined different gesture class for implementing operation on application.Various gesture supporting classes are "*QswipeGesture*", "*QPanGesture"*, "*QTapGesture","QTapAndHoldGesture*", etc. These gesture classes have many different functions,data,static data and static functions. For implementing custom gesture it is mandatory to use *"QTouchEvent"* class. It contain function like "*start ( )"*, "*update ( )", "delete ( )",* etc. to implement custom gesture.

Fig. 1 Gesture Types gives brief idea about how these gestures are to be performed on touchpad.

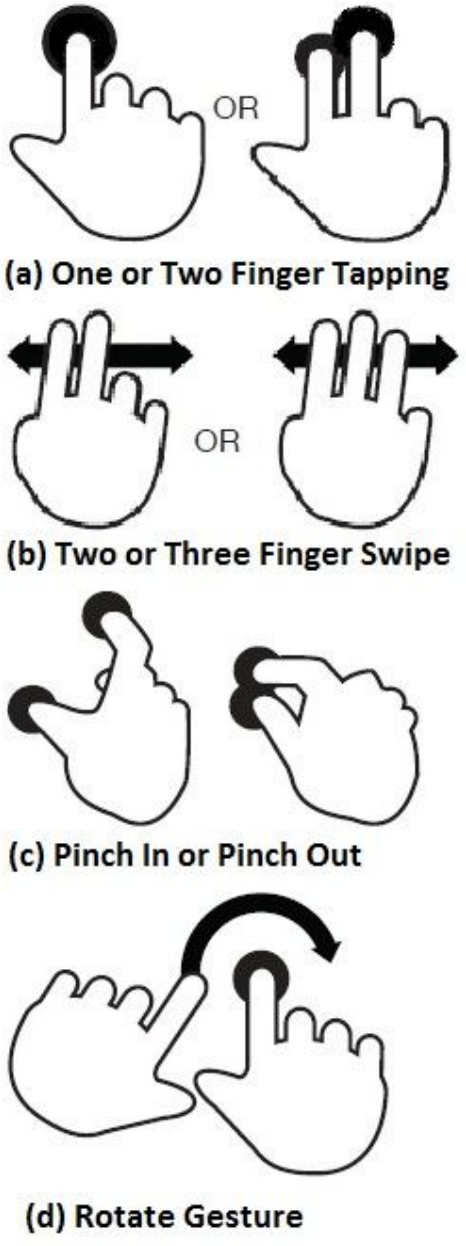

Fig. 1 Gesture Types

2. CURRENT WORK IN THIS AREA

Presently, many touchpad applications are developed but they are not supported in different operating system platform. Synaptic developed these features for windows operating system. But these softwares are not supported by other operating system Symbian and Linux. So, the platform dependency is the main problem of these softwares.

Also, for Linux operating systems (Ubuntu/Fedora), only edge-scrolling & two finger scrolling are available by default. In addition to it, two finger scrolling does not work in many flavours of Linux. Two/three finger tapping, two/three finger swipe, pan gesture are still not available in various flavours of Linux. Human Computer Interaction software company's engineers are working in this area to make these gestures available for different operating system.

3. MULTI-TOUCH SOFTWARE ARCHITECTURE

Although the platform and application may be different, the basic architecture of the Multi-touch software remains same. It consists of several different layers as shown in figure among viz. Input Hardware Layer, Hardware Abstraction Layer, Transformation Layer, Interpretation Layer, Widget Layer.

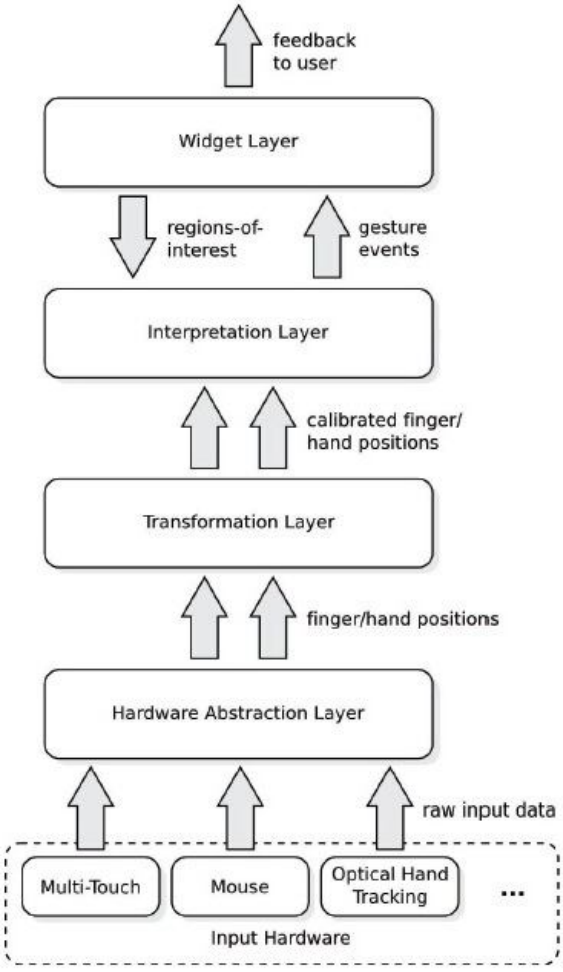

Fig. 2 MULTI-TOUCH SOFTWARE ARCHITECTURE

The *Input Hardware Layer* is the lowest layer. It generates raw tracking data in the form of electrical field measurement.

This information is then passed to *Hardware Abstraction Layer* which processes it to detect position of fingers, hands and/or objects from the raw data as per the capabilities of input device. This newly generated information is then sent to the *Transformation Layer*.

Transformation Layer converts absolute co-ordinate of device into relative co-ordinates of screen. At this stage, data provided by the touchpad is ready for interpretation.

The main task of *Interpretation Layer* is to assign some meaning to the movement of fingers performed on touchpad, i.e. gesture, using knowledge about regions on the screen. A region is a polygonal area, given in screen coordinates in which a certain set of events will be matched. Regions can be seen as a generalization of the window concept which is used in all common GUIs. For each region, a list of gestures to match is maintained. When the correct events occur within a region, the corresponding gesture is triggered and passed to the next layer.

As the mapping from motion to meaning can be expected to change for different input devices, a capability description has to be supplied which provides this mapping. This final part of our framework is the *widget layer*. Its task is to register and update regions of interest with the interpretation layer and then act on recognized gestures by generating visible output.

4. HIGH LEVEL DIAGRAM

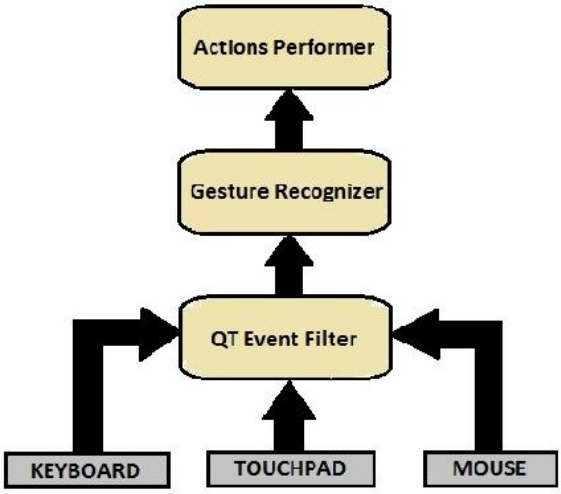

Fig. 3 Interaction of Input Devices with Application

5. WORKING AND IMPLEMENTATION

While implementing idea given in this paper first of all the image viewer's main window will be opened. The main window of this application is actually the custom *Qt Designer Class* which inherits *QMainWindow*class. *QMainWindow*has much functionality predefined in it, which are generally required by main window like *Menu Bar, Tool Bar, Status Bar, etc.* To add items in *menu bar* and *tool bar*, use of Qt Designer is the easiest way. We can also write actions, add icons, add tool tip and shortcuts for *menu bar* items using *Action Editor* of *Qt Designer.*

From *menu bar*, we have to select "Open…" option to load particular directory containing some image files. For this purpose, *QDir* and *QFile* classes are more than sufficient. From these image files, any one of the image will get displayed on the screen. Next, Previous, Zoom In, Zoom Out, Rotate, etc. these actions can be performed on the current image from *menu bar* or *tool bar.* But the main objective of this paper is to perform all these actions by making some gestures on Touchpad.

To do this, first of all, we have to enable gestures on *QLabel* object which we are using for displaying the image by passing appropriate argument to describe gesture type. This can be done by calling "*grabGesture ()"* function inside the constructor of image viewer.

e.g.

```
grabGesture(Qt::PanGesture);
grabGesture(Qt::PinchGesture);
grabGesture(Qt::SwipeGesture);
```

On generation of particular gesture, "*eventFilter()"* function will be called to detect whether it is gesture event or any other input event. If gesture event is detected then it will call "*GestureRecognizer"* to know which type of gesture is performed on current image. "*Gesture Recognizer"* class will perform the task of gesture filtering in which, it will detect which type of gesture is performed by the user and it will call the appropriate method of "*Action Performer"* class. Finally, "*Action Performer"* will contain implementations of various actions to be performed by the application such as Next, Previous, Zoom, Rotate, etc.

Now, in order to zoom the image, we don't need to drag the cursor to *menu bar->View->Zoom In or tool bar->Zoom* and then Click on it. What we just have to do is, perform the Pinch Gesture and it will automatically zoom the image, which is clearly time saving as compared to previous method.

6. FUTURE SCOPE

We have presented this paper for touch pad with support only for two/three fingers detection. But now we are planning to work with touchpad having multi touch support up to ten fingers. Due to this greatest advantage, we will be able to implement numerous custom gestures which involve up to ten fingers and maximum of the laptop functionality can be accessed only by using touchpad and gestures very speedily.

After this, we have planned to integrate multi touch gesture support in open source operating systems like Linux by working with windows session management, so that one can access the whole laptop just by using touchpad and gestures.

7. CONCLUSION

We have demonstrated the use of multi-touch gestures in accessing the laptop computers and also showed that this is the most time efficient way to access various features of different applications using image viewer application.

8. ACKNOWLEDGEMENT

We would like to express our sincere gratitude for the assistance and support of a number of people who helped us.

We are thankful to **Ms. Kavitha S.,** Department of Computer Engineering, MIT AOE, our internal guide for her valuable guidance that she provided us at various stages throughout the project work. She has been a source of motivation enabling us to give our best efforts in this project.

We are also grateful to **Prof. Uma Nagaraj,** Head of Computer Department, MIT AOE and other staff members for encouraging us and rendering all possible help, assistance and facilities.

We are also thankful to **Mr.Amar More**, Project Coordinator of Computer Department of MIT AOE for appreciating us to use Qt as our framework for implementing this paper.

9. REFERENCES

[1] JasminBlanchette, Mark Summerfield, "C++ GUI Programming with Qt 4, Second Edition"

[2] Johan Thelin, "Foundations of Qt Development"

[3] Daniel Molkentin, "The Book of Qt 4The Art of Building Qt Applications"

[4] Florian Echtler and Gudrun Klinker, "A Multitouch Software Architecture." Technische Universität München Institutfür Informatik Boltzmannstr. 3D-85747 Garching, Germany {echtler,klinker}@in.tum.de

[5] Hao Lu and Yang Li, "Gesture Coder: A Tool for Programming Multi-Touch Gestures by Demonstration." Google Research 1600 Amphitheatre Parkway Mountain View, CA 94043 yangli@acm.org

[6] Gilles Bailly, Jörg Müller and Eric Lecolinet, "Design and Evaluation of Finger-Count Interaction:Combining multitouch gestures and menus." 1Quality and Usability Lab, Telekom Innovation Laboratories, TU Berlin, Germany 2Telecom ParisTech, CNRS LTCI UMR 5141, France.