Multi-Touch Integration in Interactive Applications

Rui Nóbrega, Nuno Correia

CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
{rui.nobrega, nmc}@di.fct.unl.pt

Abstract. This paper\textsuperscript{1} presents a solution to integrate multi-touch interaction in common graphical applications. There has been extensive research on how to build multi-point, tangible or touchable hardware but most of these advances are not replicated in common standard graphics libraries. For this reason, it is harder to create applications that take advantage from the last multi-touch technologies. In this paper, a model is defined for multi-point integration and a case study is presented, where the open source graphic library Ogre 3D is extended to support multi-touch. In the model several events are suggested to be used in standard libraries, taking advantage of common spatial properties to implements the multi-point interaction.

Keywords: Multi-touch, multi-point, interaction, interfaces, computer graphics, event listeners, software integration.

1 Introduction

In the last years there has been a rise in the development of multi-touch technologies. There are innumerable new devices and techniques presented every day. This poses new questions such as how to build standard applications that take advantage of a multi-touch device? How does the multi-touch interface relate with the current common devices namely the mouse and the keyboard? Is it better to use multi-touch interaction alone or with other traditional devices?

Multi-touch or multi-point technologies introduce a new interaction paradigm by adding extra dimensions to existing single touch and single cursor devices. Current applications are, in general, conceived with the notion that there is only one cursor. The new multi-point applications require different listeners to deal with the input information. These listeners can be simply a replication of the mouse listeners or can take advantage of new properties that arise when several input points are present. These new properties include the average distance, Cartesian center, vector variation or angle variation between points.

The main contribution of this paper is to show how these properties can be used to create libraries that support more complex and interesting touch gestures in interactive

\textsuperscript{1} This work is part of the Life-Saver project which is funded by the Portuguese Foundation for Science and Technology through grant PDCT/AMB/57247/2004. We thank all the people in IMG and CITI at FCT-UNL (New University of Lisbon) for their help and support.
applications. As a case study, the graphic engine called Ogre3D [8] was extended to support multi-touch events. Using this modified version, a multi-point interface was built for a risk management application called Life-Saver[7].

The next chapter presents some related work. The chapter after explains several examples of multi-touch events useful in interactive applications. The fourth chapter has some implementation details of the prototype. In the end, there are some final remarks and directions for future work.

2 Related Work

There is extended research about multi-touch related technologies. Many of these technologies were summarized by Bill Buxton [1]. One of the main contributions for this field was the Frustrated Total Internal Reflection (FTIR) solution present by Jeff Han [3]. There are several commercial devices that support multi-touch such as Microsoft Surface [5], Apple IPhone or Apple MacBook. Other important project is the ReacTable [9] which uses a multi-touch table to interact with a music instrument application. There are essentially two main software multi-touch libraries. One is the C# API for Surface [5] and the other is ReactVision [9] from the ReactTable project. This framework follows a multi-touch software architecture [2] and implements the TUIO [4] protocol which aims to create a standard socket communication between the hardware device and the software application supporting several programming languages such as C, AS3 or java. The TouchLib[10] is a library used for finger and blob detection. Finally the MTmini [6] is a simple project that shows how to build a low cost multi-touch table.

3 Event-based approach

A multi-touch device is in its essence a multiple pointing device. As such, it has several similarities with the mouse and some major differences. First of all, in the mouse, the cursor is always present, the only change is the state of some buttons. In multi-touch, there are several cursors that might not always be present and have only one state when touching. Some devices support a different level of pressure. Most computer vision devices simulate pressure using the area of touch. Many multi-touch demos and applications take advantage of the combination of several fingers to allow the use of touch gestures, to interact with common functions of the interface.

The main idea that supports this model is that every graphical element in the application should react to certain properties that arise from multi-touch interaction. The application obtains information about the points using for example the TUIO [4] protocol. Whenever an action occurs above a certain graphical element, it is possible to derive the following direct handlers:

```javascript
onTouchEnter: { cursorID, pos}
onTouchMove: { cursorID, pos, moveVector, acelerationVector}
onTouchLeave: { cursorID}
```
These events are triggered when the user touches the device and the cursor is added, moved or removed. The `cursorID` is a sequential number generated when the user touches and destroyed when the finger is lifted. In addition to these events, there can be more complex events that are generated by different properties. On each cycle of the main loop the following properties are calculated using the detected points:

- **TouchCount**: number of points;
- **Center**: the mass center of all multi-touch points;
- **Distance**: the average distance of all points to the center;
- **DistanceVariation**: the amount of difference between the Distance in the current frame and in the last frame;
- **Angle**: given two points and the vector \( \mathbf{v} \) defined by them, the angle formed by \( \mathbf{v} \) in the current frame and \( \mathbf{v} \) in the last frame.

Given these properties, operations such as zoom and rotate can be implemented with the following triggers:

```plaintext
if: TouchCount ≥ 2 and DistanceVariation > 0
  onDistanceChange: { allCursorID, allPos, Center, Distance, DistanceVariation }

if: TouchCount = 2 and Distance > 100px and Angle > 0º
  onRotation: { curID_1&2, pos_1&2, Center, angle }
```

When using these triggers, it is important to take into consideration that they will sometimes interfere with each other. A small distance change between two very near points can lead to a large rotation due to the imprecision of the touch. For compatibility with the current interfaces some of the mouse triggers can be emulated. For example, the `onMouseMove` event can be triggered when the screen is touched by one finger and it moves. The `onMousePressed` would work when the user tapped the screen with other finger near the first. The `onMouseDrag` would be similar to the `onMouseMove` but with two fingers.

A useful event would be one suitable for dual touch. In many applications, the user wants to hold an object in two points and control it by moving those points around. With a combination of the move vector of the first point, its distance variation to the other and the angle made between the vector uniting the two points and the horizontal axis, it is possible to calculate translation, scale and rotation and deliver it to the developer in the following trigger:

```plaintext
if: TouchCount = 2
  onDualMove: { curID_1&2, pos_1&2, translation, scale, rotation }
```

Using this method it is possible to create the illusion that the object is being directly manipulated. Several other properties could be mapped, such as the area of the polygon formed by all points or the amount of time the user is touching on the same spot. For simplicity, the pressure was left out of each listener but it is also an important dimension.

### 4 Graphic Library Extension

To test the feasibility of these handlers an extension was made to a known open source graphic library called Ogre3D [8]. To create an application using the multi-
touch extension the user has only to create a class that implements an abstract class of the handlers and submit that class as a listener of multi-touch events. To evaluate this implementation, a prototype application was modified to support multi-touch and a simple low-cost multi-touch table was built to test it.

The prototype is an object-oriented and event-driven C++ application. The prototype application (Fig.1) is a disaster management tool designed to visualize flood emergencies. The multi-touch adaptation of this application is centered on the navigation system. The user can navigate through the map using the following terrain operations: drag, zoom, pitch and rotate.

5 Conclusions and Future Work

This paper describes a event-based solution to the multi-touch software integration problem. Several studies are being made in order to improve and evaluate it. The event-driven approach proved to be functional and scalable, reducing the gap for multi-touch integration for the developer. In the future, we intend to explore different touch gestures to interact with applications and we are interested in extending the library to support interaction with multiple objects at the same time.

References

9. ReactTable, http://mtg.upf.edu/reactable/