A Multitouch Software Architecture

Florian Echtler, Gudrun Klinker
{echtler,klinker}@in.tum.de
Technische Universität München
Institut für Informatik I16

Multitouch is everywhere..

.. in terms of hardware, at least. However, the software side is still lacking, especially in terms of interoperability. We present a layered architecture which aims to subsume the properties of already existing applications and integrating them with each other.

This approach offers two benefits:

- existing software can be moved between different hardware platforms more easily
- developers can use a single API to access a variety of multitouch systems, don't have to „reinvent the wheel“

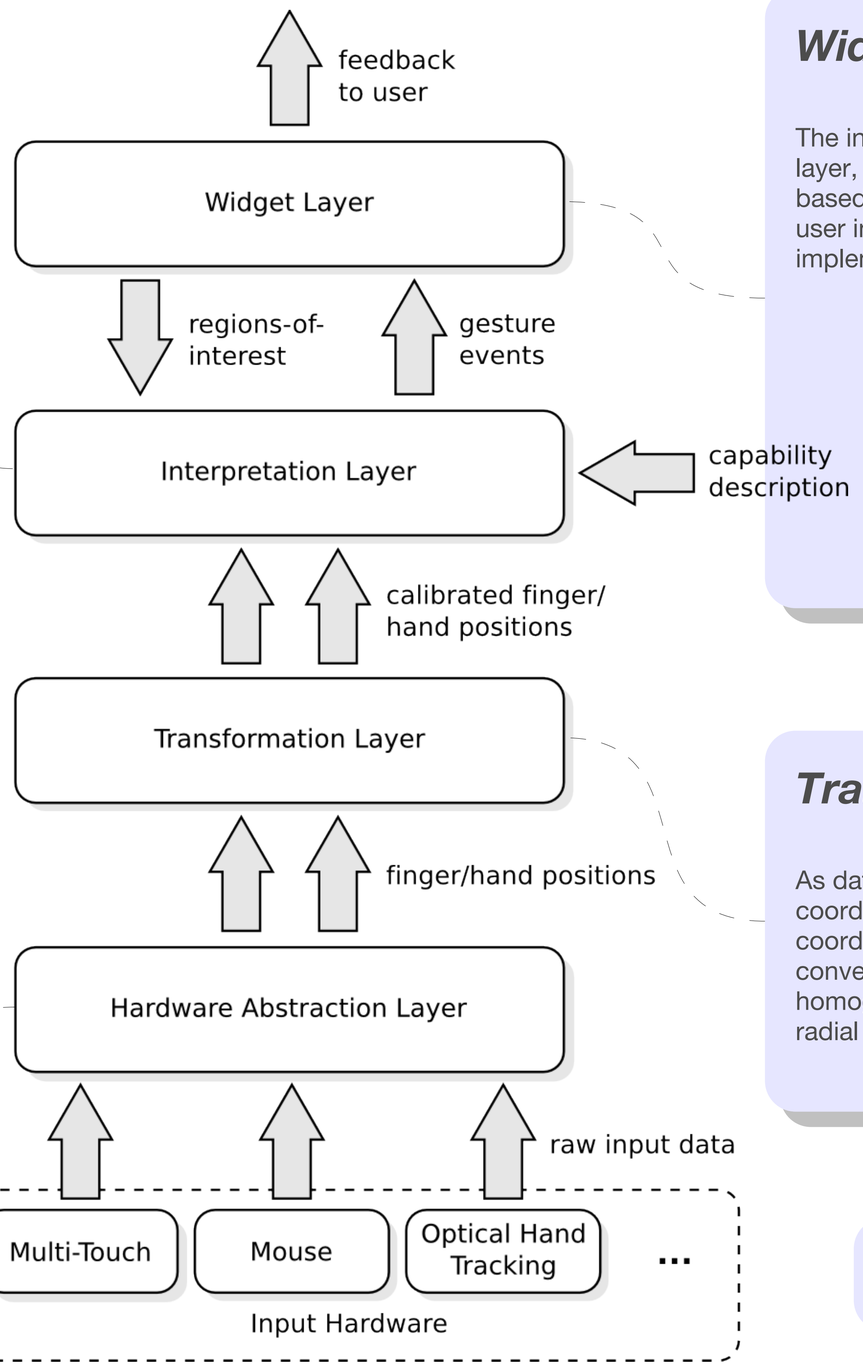
Examples:

- FTIR
- SMARTBoard
- DiamondTouch
- SmartSkin
- ART Handtracking
- Multi-Mouse
- iPhone
- ....

Interpretation Layer

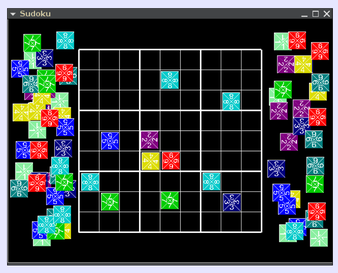
At this point, the data still has no meaning beyond positions in screen space. The interpretation layer changes this. It introduces the notions of regions, events and features.

- *Regions* are a generalization of the ubiquitous window concept commonly used in today's GUIs. They are described by closed polygons and ordered in a stack.
- *Events* are triggered through the user by performing certain actions within a region. The application can either select from a list of predefined events („move“, „tap“ etc.) or compose a new one.
- *Features* are the building blocks of events. Every feature describes a single, atomic property of input data, such as the overall motion vector or the centroid of all finger contact spots. When all features for a certain event match, this event is delivered to the next layer.



Widget Layer

The interpreted events are passed on to the widget layer, which then acts on them. Our implementation is based on OpenGL and provides rotation-independent user interaction. There can also be other implementations of this layer, e.g. a Qt-based variant.



Transformation Layer

As data from the hardware is usually given in a device coordinate system, it has to be converted to screen coordinates. The transformation layer accomplishes this conversion by applying, e.g., a pre-calculated homography to the data. For vision-based tracking, a radial undistortion step may also be performed.

Hardware Abstraction Layer

The multitouch devices which have been in use in recent years consist of such various hardware as DiamondTouch, FTIR-based systems. Even systems that track the user's hands in 3D could be used. This layer takes the data from the underlying hardware and converts it into a common format with information about finger, hand and possibly also object positions. TUIO data can also be used through a converter.

Get it at http://tisch.sf.net/

Future Work

- Define standard set of events
- Define event-to-feature mappings for common hardware devices
- Create HAL implementation for other types of input hardware

- Extend OpenGL-based widget set with standard items (Button, Slider..)
- Create adapters for existing widget libraries (e.g. Qt, FLTK, ...)

- Create end-user applications
- Refine architecture based on requirements

References

- J. Han. *Low-cost multi-touch sensing through frustrated total internal reflection.* UIST ’05.
- P. Dietz & D. Leigh. *DiamondTouch. A multi-user touch technology.* UIST ’01.
- J. Rekimoto. *SmartSkin: an infrastructure for freehand manipulation on interactive surfaces*. CHI ’02.
- F. Echtler et al. *Shadow tracking on multi-touch tables.* AVI ’08.
- R.A. Diaz-Marino et al. *Programming for multiple touches and multiple users: A toolkit for the DiamondTouch hardware.* UIST ’03.
- M. Kaltenbrunner et al. *TUIO: A protocol for table-top tangible user interfaces.* GW ’05.
- C. Shet et al. *DiamondSpin: an extensible toolkit for around-the-table interaction.* CHI ’04.